A COMPARISON OF TWO MICROCOMPUTER DATABASE MANAGEMENT SYSTEM PRODUCTS

by

Gary A. Radke

B.S., Kansas State University, 1979

A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by :

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

The software market is currently crowded with microcomputer based database management systems. The complexity of the systems available ranges from simple information management systems which are inflexible in their format to the advanced systems with programmable products that allow the user to easily write an application tailored to their precise needs. The simple systems force the user to adapt their application to the structure of the database management system; the advanced systems often do not require the user to learn a great deal more about database management systems. The reviews of these -database systems typically consist of a table that lists the various products on one axis and categories of system features on the other axis with a mark denoting whether or not a product supports a given feature. These tables are frequently accompanied by statistics such as the number of database files that can be kept open at one time, the possible number of records per file, the time required to index a given number of records and other similar information meant to allow a potential user to decide which product would best meet their needs. Unfortunately, comparisons of these products is seldom made using the implementation of a real life business application and thus having the potential of providing information on the relative ease of implementation of a front end for these products and the useability of the final product. This report will attempt to provide

a qualitative comparison of two of the more advanced computer database management systems.

At the time this report was initiated (1986) the two most favorably reviewed computer based, relational, programmable database management systems on the market were dBASE III+ and RBASE 5000. A review of their technical specifications showed them to be fairly comparable in system requirements and features offered so these two systems were chosen for comparison by implementation.

DBASE III+ is a product of Ashton Tate Inc. It is a single or multiuser relational database management system with the minimum requirements of 256 K memory and two disk drives. A database created with dBASE III+ may be manipulated from the assist mode which provides a series of menus that guide the user through the steps necessary to create a database file, add records to the database, delete records from it or to browse through and modify existing database records. The assist mode may also be used to build queries for the database. These tasks may also be performed from the "dot prompt" without the aid of the assistant. An editor is also provided to take advantage of the programming capabilities of the system.

RBASE 5000 is a Microrim product. It too is a single or multiuser system with the minimum system requirements of 320 K of memory and a hard disk. RBASE 5000 provides a prompt mode, similar to the assist mode of dBASE III+, that guides the user through database creation and manipulation and it provides an editor for use when creating custom applications.

To provide a basis of comparison for these two systems, an

application was needed that would both benefit from implementation with a database management system and was sufficiently large to provide information on the effort required for the implementation. The application also had to be small enough so as to not require an unreasonable amount of time to complete the implementation. Another desirable characteristic of the enterprise to be considered was that the end users of the application should not be computer professionals, thus requiring that the user interface allow database manipulation with little chance of the database integrity being compromised. These criteria were met by the Peoples' Grocery Cooperative Exchange which was the enterprise chosen for implementation. The operation of Peoples' Grocery is described elsewhere in this report.

The implementation of the application was accomplished in the following manner. The logical design of the database was developed and this logical design was then implemented in both dBASE III+ and RBASE 5000. The aim was for the final application of each product to perform essentially the same function. In this way the effort necessary to product the application could be evaluated and the resulting user interface could be compared. It should be noted here that the RBASE 5000 package used is a promotional package and as such it contains all of the standard RBASE 5000 features, commands and functions but it is limited in the number of tables and records. Even though it was limited in this way the package was adequate to complete the task at hand.

This report will discuss the creation of an extensible, "user

friendly" front end for the database of a small enterprise. A user interface has been developed using both dBASE III+ and RBASE 5000 with the object of providing qualitative information on the effort required to establish the database and produce a user interface for each of these systems.

This chapter has discussed the need for practical information when evaluating a database management system. Chapter 2 will describe the enterprise to be implemented. Chapter 3 will cover the operation of the application developed in dBASE III+ while chapter 4 will provide similar information for the application developed in RBASE 5000. Chapter 5 will compare the development of the two applications and provide comments on various aspects of the process. Chapter 6 will review both database management system implementations and provide suggestions for future study.

Chapter 2

Problem Environment and Database Design

Introduction:

Among the problems faced by a small enterprise that wishes to implement and maintain a DBMS as part of its operation is that the people actually manipulating the database may not fully understand the results of their actions on the database. Though the database may be well designed, indiscriminate entry of new records and access to all records by an inexperienced user may result in operational chaos for the enterprise. To alleviate this potential problem, it may prove useful in many instances to provide a front end to a given DBMS which is customized to accommodate the needs of an enterprise. This front end should meet the operational requirements of the enterprise while preventing misuse of the database.

This chapter will describe the operation of a small enterprise and the design of the database, and the user interface used to satisfy the requirements of the enterprise.

Enterprise Description:

Peoples' Grocery Cooperative Exchange is a member owned food buying club that also operates a small retail grocery store in Manhattan, Kansas. Its business is conducted in the following manner.

Every two weeks an order is placed to the grocery's main supplier. At this time members have the opportunity to submit an

individual order in which they may order a full or a partial case of an item. If they order a partial case, other members, or the store, must order the balance of the case before the given item may be ordered. When all of the individual orders have been submitted, they are compiled into an order which is placed to the supplier. Between the time the order is placed and the time it is delivered, labels are prepared by hand which detail the catalog number and description of an item, who ordered the item and how much of the item each member ordered. When the shipment is delivered, it is distributed among members in accordance with the prepared labels. At this time note is taken of items which were not delivered or which were refused due to damage in transit or substandard quality. Adjustments reflecting these undelivered goods are made to the orders involved. Finally, incidental costs such as shipping, markup, taxes and member equity are added to the base price of the items received, and this bill is presented to the participating member for immediate payment.

After members have received their orders they may remove them for home consumption or they may sell all or a portion of their order by consignment through the groceries retail storefront.

To do this the member repackages the items they wish to sell, prices them and identifies each package with her/his member identification number. The number, price, and description of the item are recorded along with the identification number of the member selling the item. As the items are sold, their sale is recorded and when all of a given item has been sold, the consigning member is paid the price she/he requested.

Providing members the opportunity to sell merchandise on consignment is not without cost. Rental of the retail space, utilities, equipment, repairs, and the salaries of the manager and cashier are all expenses which are paid for by the markup the grocery adds on to the cost of the goods sold to the members and through the retail storefront.

Given this mode of operation, there are two areas in which a database management system would be helpful.

First, the effort involved in compiling, submitting and distributing any order is done by a frequently changing cast of volunteers. Some of the work is tedious and error prone which may prompt more frequent turn over of volunteers willing to devote their time to complete the necessary work. Thus, easing this task may keep them enthused and willing to participate.

Secondly, the business is small and while expenses remain fairly constant, the amount of sales may vary considerably from month to month. For this reason, some outstanding debts or members who are owed money for consignments which have already been sold may remain unpaid so the grocery can maintain a positive bank balance.

To avoid serious financial difficulty, it would prove useful to have the ability to assess the gross financial situation of the grocery with relative ease so that prudent decisions can be made in a timely manner.

A database to be used for the purpose of addressing these needs has been implemented using Dbase III plus, a product of Ashton Tate, and Rbase 5000, a microrim product. Both are relational database

products designed for use on personal computers.

Logical Design:

The logical design for the database needed to accomplish the work necessary is given initially through the entities/objects of concern given in figure 2.1

Each of the entities will be implemented as relations in the Dbase III + and Rbase 5000 systems. These relations were checked for normal form and they are in 3NF.

Summary:

The aim of the system supporting the database is to maintain enough information so as to allow removal of some of the burden of order preparation and distribution through a user friendly system that virtually anyone can use after a small amount of instruction. The system must insure that when information is entered all of the information necessary for proper system operation is entered. It must also ensure that an inexperienced user does not corrupt the database by inadvertently removing existing records or by entering duplicate records. Thus, record removal is left to the system administrator and entry of duplicate records is not allowed. The next chapter further describes the user interface and system implementation.

MEMBER
Member id, First name, Last name, Street address, City, State, Zipcode, Home phone, Work phone: Key Member id.

SUPPLIER
Supplier name, Street address, City, State, Zipcode, Phone number, Contact name: Key Supplier name.

PRODUCT CATALOG
Supplier name, Product identification number, Description, Case price, Unit price, Shipping weight: Key Supplier name, Product identification number.

MEMBER ORDER
Member id, Supplier name, Product identification number, Date, Quantity
:Key Member id, Supplier name, Product identification number, Date.

CONSIGNMENTS
Member id, Description, Date, Price, All sold, Member paid:
Key Member id, Description, Date, Price.

EXPENSES
Date, Amount, Description, Paid to, Paid: Key Date, Amount, Description, Paid to.

Figure 2.1 Entities of the system

Chapter 3

Dbase III+ Implementation

3.0 Introduction:

The overall goals of this system are twofold. First it aims to aid Peoples' Grocery in the completion of time consuming, frequently performed, error prone tasks by reducing the effort required to get the work done, while reducing the chance for errors. Secondly, it is meant to give the management the ability to assess the general fiscal health of the operation on short notice.

To do this, an interface with dBASE III + was created which allows a naive user to enter data into the database without inadvertently corrupting it. Additionally, the user is able to generate reports of a predetermined format with a minimum amount of effort and a reduced chance for error.

The system is driven by a main menu from which the user can initiate any of the tasks the application can perform (figure 3-1). Options in the menu fall into the broad categories of data entry/update and report generation. With the options that appear in the data entry/update section the user is allowed to enter information for members, order information, supplier data, product data, consignment information and store expense information. From this set of options the user may also make changes to selected fields in existing records. Data entry is accomplished through the use of forms which prompt the user for input. The system checks for and disallows duplicate forms and it requires that key fields in

each form have data entered in them. Report form generation is generally accomplished by the user answering yes or no to a system prompt or by entering a date. The options in the reports section allow the user to print product breakdown labels, order summaries, bills for individual member orders as well as reports of consignment status and reports summarizing expenses. To accomplish all of the work done by the application required 2500 lines of code in 99 procedures. The rest of this chapter is devoted to describing in more detail the actions taken by the system when options are chosen from the main menu.

3.1 Common Actions:

There are actions which are commonly and repeatedly performed in various parts of the program; these will be described in this section here and referred to later in the text.
These are 1)locate a member using the member identification number, 2)browse the file of current members, and 3)browse the file of current suppliers.

3.1.0 Locate a member using the member identification number:

The purpose of this action is to identify an existing member. The user is prompted to enter a member's identification number (figure 3-2). If the number entered does not exist in the database, an error message is displayed and the user is allowed to try again . If the number entered does exist, the appropriate data for the action to be performed is retrieved.

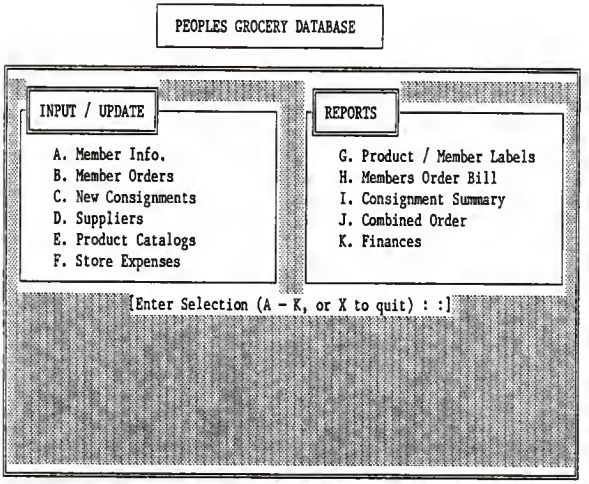

Figure 3.1 dBASE III+ Main Menu

```
ENTER MEMBERS NUMBER          0
```

Figure 3-2 Prompt for Member Identification number

3.1.1 Browse the file of current members:

On occasion, the user may not know a given member's identification number. The browse member file action allows the examination of identification numbers and their associated names. The names and identification numbers are displayed along with three user response options (figure 3-3). By typing 'Y' the user is choosing to perform an action involving the member displayed. By typing 'N' the user is choosing to not take an action involving this member. With this response, another member name and number is displayed and the user is again allowed to respond. In this way all member names and identification numbers will be displayed in ascending order of identification numbers. When the entire file has been examined, the user is informed and returned to the previous menu. The third possible response is 'X' which abandons the browsing operation and returns to the previous menu.

3.1.2 Browse the file of current suppliers:

Some actions require identification of a supplier. To avoid the spelling errors possible when entering a supplier's name, names of the suppliers are displayed one at a time and the user is presented with the same response options as in the browse member file (figure 3-4). The actions taken for a given response are also the same only of course associated with the supplier instead of a member.

```
MEMBER #    1        NAME DON HALEY

UPDATE MEMBERSHIP INFO FOR THIS MEMBER? Y/N
```

```
Y - UPDATE INFORMATION FOR THIS MEMBER.

N - SKIP THIS MEMBER, LOOK AT THE NEXT.

X - RETURN TO THE PREVIOUS MENU.
```

Figure 3-3 Prompt to Browse Membership File

```
SUPPLIER  BLOOMING PRARIE

UPDATE INFO FOR THIS SUPPLIER? Y/N
```

```
Y DISPLAYS INFO FOR THIS SUPPLIER

N SHOWS YOU THE NEXT SUPPLIER

X RETURNS YOU TO THE PREVIOUS MENU
```

Figure 3-4 Prompt to Browse Supplier File

3.2 Entering and updating information:

Selection of an action in this set of options (figure 3-1) allows the user to enter or update information for the following entities, Member Information, Product Catalog, Supplier Information, Member Orders, Consignments, and Expenses.

Member information:

When this action is chosen it displays a menu (figure 3-5) allowing the user to add a new member, update an existing member record or return to the main menu.

Adding a new member: This action displays a form (figure 3-6) with spaces for an identification number, first name, last name, street address, city, state, zip code, home phone number and work phone number.

The module requires the entry of an identification number and a first and last name. Failure to enter this information results in an error message and then returns the user to the form entry mode. It also checks for and disallows duplicate identification numbers. Once the form has been completed the user is allowed to store the information as entered, go back and make changes, or abandon the operation without storing any data. Acceptance or abandonment of the operation returns the user to the membership information menu.

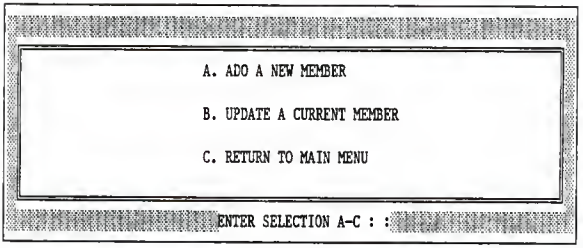

Figure 3-5 Membership Information Menu

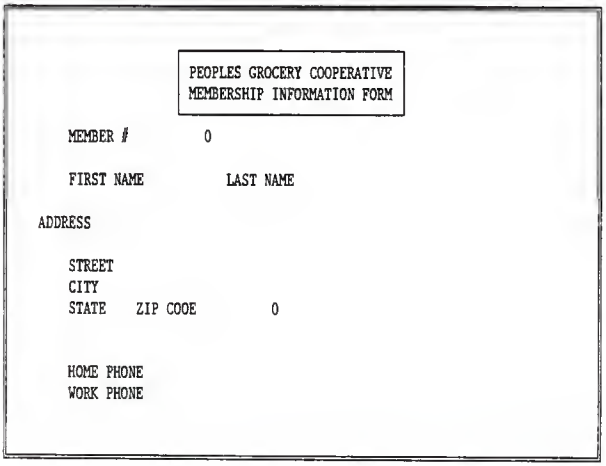

Figure 3-6 Membership Information Form

Updating an existing member: When this action is chosen, it displays a menu that allows the user to enter a known identification number, as described in section 3.1.0, browse the member file a record at a time, as described in section 3.1.1, or return to the previous menu.

When a member has been identified, the member information form (figure 3-6) is displayed, and the user may make changes to the address and phone number fields. As before, upon completion of the changes, the user is allowed to save the changes, return to the form to make corrections or abandon the operation without saving the changes. After acceptance or abandonment, the user is returned to the previous menu.

Member Order:

When this action is chosen, a menu (figure 3-7) is displayed allowing the user to enter a new order or update an existing order.

Entering a new order: The supplier to which the order is to be placed, and the member placing the order are selected as described in sections 3.1.1 and 3.1.2 respectively. When these tasks have been successfully completed, an order form is displayed (figure 3-8) in which the user enters the product identification number. If the number entered doesn't exist for the chosen supplier an error message is displayed and the user is allowed to reenter. After a valid product identification number has been entered, the user may enter the number of units

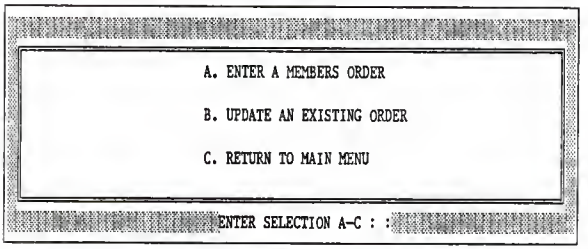

Figure 3-7 Member Order Menu

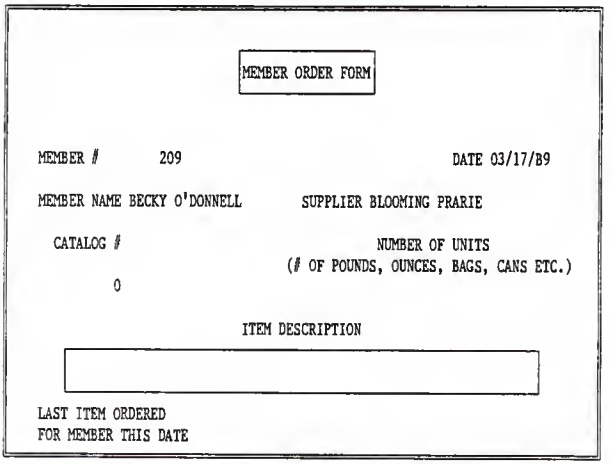

Figure 3-8 Member Order Information Form

the member wishes to order. The only other items of information stored with the order are the supplier name, member identification number and the order date. The supplier and member have already been identified and the date is taken to be the system date. When the form is complete, it can be saved, corrected, or abandoned. If saved, the user may order another item for the chosen member or choose a new member for whom to place an order.

Update an existing order: A supplier and member are identified, e.g.; when submitting a new order. The order form is displayed (figure 3-8) and the user is allowed to enter the date of the order to be updated. If the chosen member hasn't placed an order with the chosen supplier on the date entered, an error message is displayed. Then they are allowed to reenter the date. When a valid date has been entered the user enters the identification number of the ordered item for change. If the item hasn't been ordered by the given member on the given date, an error message is displayed and the user has the option of abandoning the operation or trying again. When an ordered item has been successfully identified and the desired changes made, the user is given the option of changing another item for this supplier and member. If no further changes for this member are needed, a change of order for another member or return to the enter / update menu is possible.

Supplier Information:

When this action is chosen, it displays a menu (figure 3-9) allowing the user to add a new supplier, update an existing supplier record or return to the main menu.

Add a new supplier: When this action is chosen it displays the supplier information form (figure 3-10) in which may be entered the supplier name, street address, city, state, zipcode, phone number and the name of the supplier's representative with whom the grocery has contact. A value must be entered for the supplier's name but entry of all other values is optional. Upon completion of the form, if no value has been entered for the supplier's name an error message is displayed, and the user is returned to the form. This is also the case if the supplier name already exists in the database. After completion, the form is stored, or abandoned as described in the member information section.

Update an existing supplier: When this action is chosen,it prompts the user to browse through the supplier file as described in the section 3.1.2.

When the proper supplier has been identified, the current information for that supplier is displayed (figure 3-10) and the user may update all of the fields except the supplier name. The changes may be accepted, changed again or the whole operation abandoned as previously described.

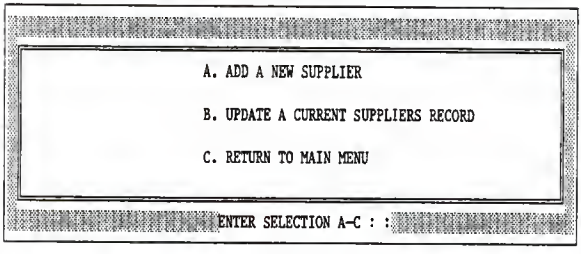

Figure 3-9 Supplier Information Menu

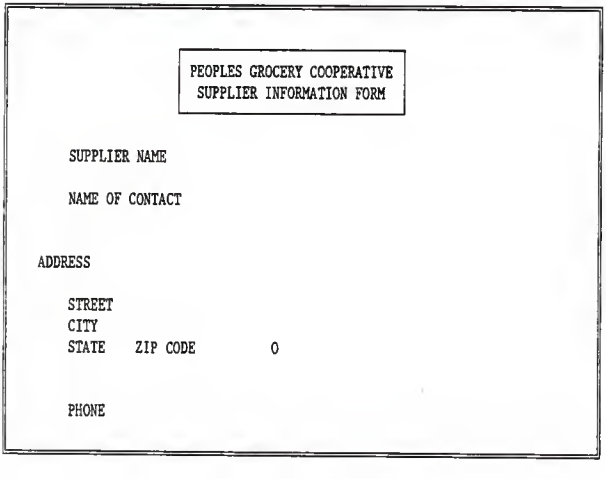

Figure 3-10 Supplier Information Form

Product Catalog:

When this action is chosen it displays a menu (figure 3-11) allowing the user to add a new product to a given supplier catalog, update product information or return to the main menu.

Add a new product: When this action is chosen it displays a form (figure 3-12) with spaces for the product identification number, supplier name, product description, shipping weight, case price, and unit price.

The module requires that every space in this form have an entry and it checks for and disallows duplicate identification numbers for the same supplier. Upon completion of the form, the user may save the information abandon the operation with no change to the database or return to the form to make corrections.

Saving the form or abandonment of the operation returns the user to the enter/update menu (figure 3-11).

Update an existing product: When this action is chosen, the user is asked to select a supplier as described in section 3.1.1. When a supplier has been chosen for product update, a menu is displayed (figure 3-13) allowing the user to enter a known product identification number.

Enter identification number: When this action is chosen it allows the user to enter a product identification number (figure 3-14). If the number is not present in the

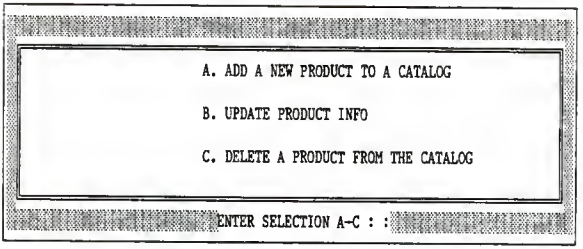

Figure 3-11 Product Catalog Menu

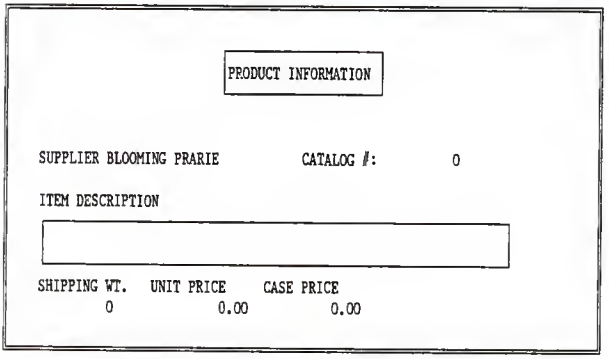

Figure 3-12 Product Information Form

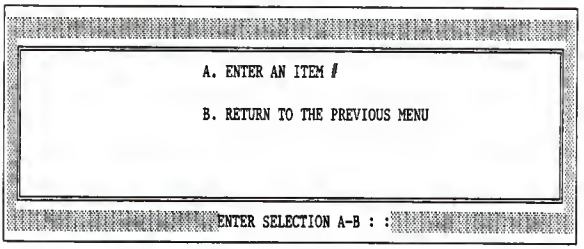

Figure 3-13 Product Update Menu

UPDATE CATALOG FOR PRODUCT # 0

Figure 3-14 Product Selection Prompt

chosen supplier's catalog, the user is informed and asked to reenter the number. If the number is present, the product form is displayed (figure 3-12) and the user may make changes to the shipping weight, case price and unit price fields. When changes are complete, they may be saved, the operation abandoned, or the user may return to the form to make corrections if necessary.

When the changes are saved or abandoned, the user is asked if they want to make other changes in this supplier's catalog. To do this, another product identification number is entered and the process is repeated. After all updates have been completed for the current supplier, a new supplier can be chosen and the update process repeated, or the user may choose to return to the enter/update menu (figure 3-1).

Consignments:

When this action is chosen it displays a menu (figure 3-15) allowing the user to enter a new consignment or update existing consignments.

Enter a new consignment: When this action is chosen it displays the consignment information form (figure 3-16) allowing the user to enter the member identification number, date, consignment description, the number of items placed on consignment, the price to be paid to the member for each item,

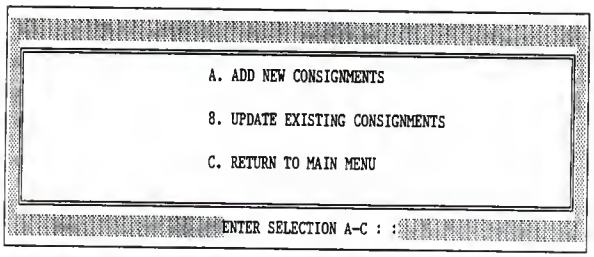

Figure 3-15 Consignment Information Menu

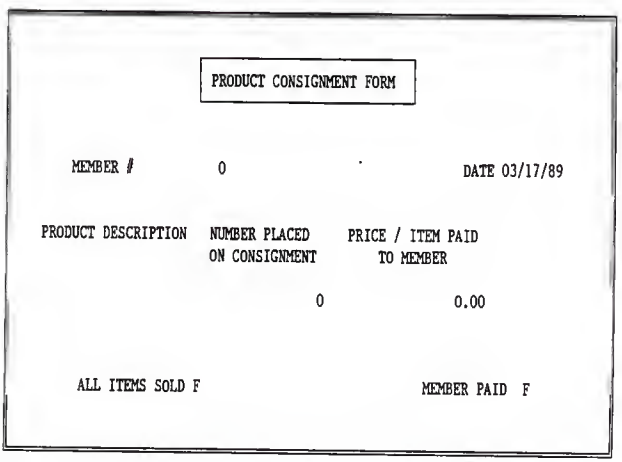

Figure 3-16 Consignment Information Form

a field that designates when all of the items have been sold, and a field that designates if the member has been paid for these items. For this form the date is taken to be the system date but it may be changed. All fields except "all sold" and "member paid" must have an entry. An error message is displayed if any fields are left blank and the user is returned to the form. Duplicates are checked for and disallowed.

When the form has been completed, the user may go back and make changes to it, save it or abandon the operation with no change.

Update an existing consignment: Allows the user to enter a known member as described in section 3.1.0 identification number or browse the member file as described in section 3.1.1. When a member has been identified, the user is allowed to browse through the items this member has on consignment in a manner similar to browsing the member file. The member name, consignment, date and consignment description are displayed along with a user response menu. Typing 'Y' allows the user to change the number of items on consignment, the price of each item, the "all sold" field, and the "member paid" field. When changes have been completed, the form may be saved, modified further, or abandoned with no changes made. By typing 'N'the user is shown the next consignment in the members file. When all consignments for this member have been displayed, the user is notified and returned to the choose member identification

number menu. Typing 'X' returns the user to the choose member identification number menu.

Expenses:

When chosen this option displays a menu (figure 3-17) that allows the user to enter a new expense form or to browse and update existing forms.

Enter an expense: When this action is chosen it displays an expense form (figure 3-18) that allows the user enter the date, the amount of the expense, who was paid, a description of the expense and whether or not the bill has been paid. All fields but the "paid" field must have an entry and duplicate fields are checked for and disallowed.

When the form has been completed it may be saved, modified, if necessary, or abandoned without being saved.

Browse the expense file: This action displays existing expense forms (figure 3-19) beginning with the most recent, and a user response menu. Typing 'Y' retrieves the current form and allows the user to change the boolean expense paid field. Typing 'N' causes the next record to be displayed, and Typing 'X' abandons the browsing operation and returns the user to the previous menu.

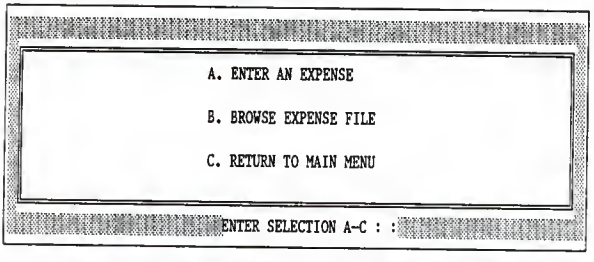

Figure 3-17 Expense Information Menu

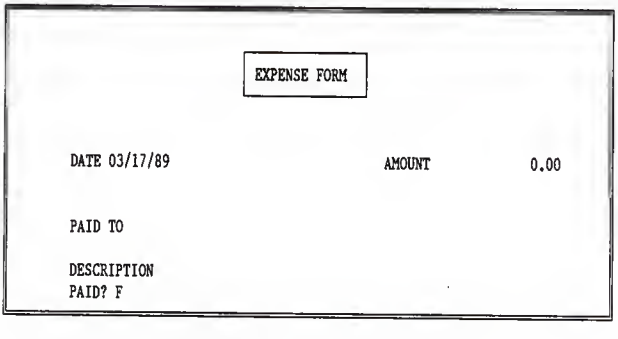

Figure 3-18 Expense Information Form

3.4 REPORTS

There are reports that are used routinely during the operation of the grocery e.g., product breakdown labels, member order bills and combined order summaries. These forms must be prepared whenever the grocery submits an order to a supplier. In addition to these reports, there are others that are required on a sporadic basis; consignment summaries and reports regarding the stores expenses may be requested at any time. The actions the user must take to generate these reports are described in this section.

Product breakdown label:

When this action is chosen it displays a menu (figure 3-20) allowing the user to print product breakdown labels or return to the main menu.

Print product breakdown labels: When this action is taken, a message (figure 3-21) is displayed asking if creation of labels for the order placed to the displayed supplier on the displayed date is desired. By typing 'Y' labels are printed showing the date of the order, the catalog number of the item and its description. The rest of the label consists of the identification number and name of each member that has ordered this item as well as the quantity they ordered (figure 3-22). When labels for all ordered items have been printed, the user is returned to the main menu.

By typing 'N' at the prompt in figure 3-21 another supplier

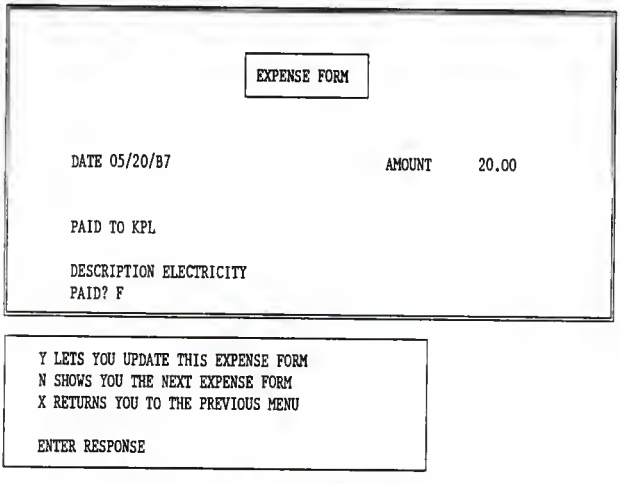

Figure 3-19 Prompt to Browse Expense File

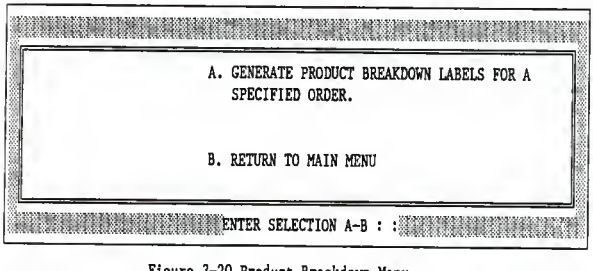

Figure 3-20 Product Breakdown Menu

```
PRINT LABELS FOR ORDER SUBMITTED TO BLOOMING PRARIE ON 02/21/89
Y/N ?
```

Figure 3-21 Product Breakdown Label Prompt

```
                    PRODUCT BREAK DOWN LABEL
ORDER DATE 02/21/89
CATALOG # 1001
DESCRIPTION WHEAT BERRIES
-------------------------------------------------------------

MEMBER #          MEMBER NAME          QUANTITY

   2              JAYNE LINK             5.00
  209             BECKY O'DONNELL       50.00
Press any key to continue...
```

Figure 3-22 Product Breakdown Label

name and date combination is displayed. When all possible order date and supplier combinations have been examined, the user may reexamine the list or return to the main menu.

Order bill:

When this action is chosen it displays a menu (figure 3-23) allowing the user to continue with the operation or return to the main menu.

Print order bill: When this action is chosen, the appropriate date and supplier are chosen as described in section 3.1.2 and the user is asked if any items were not delivered or were refused when the order arrived. If this case exists, the catalog numbers of those items are entered and the items are deleted from the order. Otherwise, the user is asked to enter some information that varies from order to order. These are shipping cost as price per pound, member equity charge sales tax rate, price markup rate and discount. At this point, the user may go back and make corrections, abandon the operation or continue. If the continue option is chosen, a bill is printed (figure 3-24) for each member participating in this order. Each bill lists the members name, identification number, the suppliers name and the order date plus the catalog numbers descriptions, quantity ordered, unit cost, and total price for each item this member has ordered. Any adjustments as entered above are also listed along with the total cost to the member.

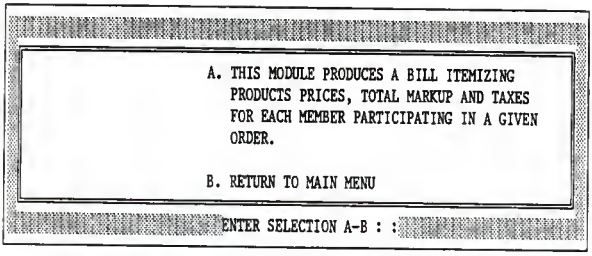

Figure 3-23 Order Bill Menu

ORDER BILL FOR JAYNE LINK MEMBER # 2

SUPPLIER BLOOMING PRARIE ORDER DATE 02/21/89

CATALOG #	DESCRIPTION	QUANTITY ORDERED	UNIT COST	TOTAL PRICE
1001	WHEAT BERRIES	5.00	0.50	2.50

SUBTOTAL		2.50
MARKUP		0.38
DISCOUNT -		0.00
TAX		0.16
EQUITY		0.00
TOTAL SHIPPING COST		0.00
TOTAL COST	$	3.03

Press any key to continue...

Figure 3-24 Order Bill Report

Consignment Summary:

When this action is chosen it displays a menu (figure 3-25) that gives the user the option to produce one of three consignment reports these are sold unpaid consignments for a given member, all members with sold unpaid consignments, all members with sold and unsold unpaid consignments.

List sold unpaid consignments of a given member: This action lets the user identify a member via the actions described in sections 3.1.0 or 3.1.1. If the chosen member has any consignments which have been sold but for which they have not been paid, the consignment descriptions and the values of those consignments are listed, as well as the sum value of all sold, unpaid consignments for this member (figure 3-26). If the chosen member has no sold, unpaid consignments, a message to that effect is displayed and the user is allowed to continue browsing the member file or abandon the operation and return to the main menu.

List all members with sold unpaid consignments: This report lists each member that has items on consignment which have been sold but for which they have not been paid by the grocery. The total amount owed to each member for their sold consignments is listed along with the total owed to all members for sold consignments (figure 3-27).

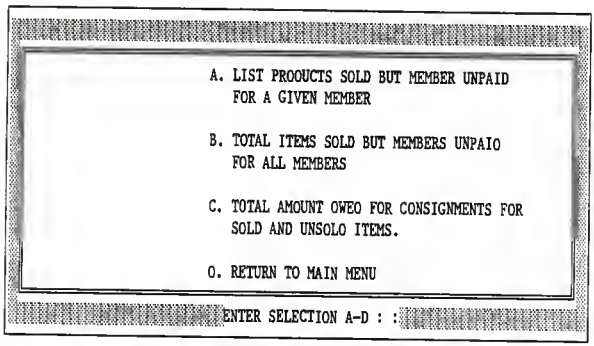

Figure 3-25 Consignment Reports Menu

SUMMARY OF CONSIGNMENTS SOLD BUT FOR WHICH MEMBER HAS
NOT BEEN PAIO

MEMBER # 209 MEMBER NAME BECKY O'DONNELL

DESCRIPTION	UNITS	UNIT PRICE	CONSIGNMENT VAL
BROWN RICE	3	1.00	3.00
GARLIC	10	0.25	2.50
GOOSE EGGS	10	0.50	5.00
TOTAL AMOUNT OWED THIS MEMBER FOR SOLO ITEMS			10.50

Press any key to continue...

Figure 3-26 Sold, Unpaid Consignments Summary for a Given Member

List all members with sold and unsold unpaid consignments: This report lists each member having items on consignment and the total amount owed to them for both sold and unsold consignments as well as the total owed to all members for consignments (figure 3-28).

Combined order:

When this action is chosen it displays a menu (figure 3-29) allowing the user to print the combined member order or return to the main menu.

Print combined member order: When this action is chosen, a supplier is identified as described in section 3.1.2, an order date is chosen, and a report is printed which contains the supplier name, date of the order, catalog numbers descriptions number of cases , case price and total cost for each item ordered (figure 3-30). When all items have been listed, the total value of the order is given and the user is returned to the combined order menu.

Finances:

When chosen, this action displays a menu (figure 3-31) giving the user the option to produce three reports. These reports are list

SUMMARY OF SOLD, UPAID CONSIGNMENTS FOR ALL MEMBERS

MEMBER #	MEMBER NAME	AMOUNT OWED
4	HUGH KNOX	100.00
209	BECKY O'DONNELL	10.50
TOTAL AMOUNT OWED TO ALL MEMBERS FOR SOLD CONSIGNMENTS		110.50

Press any key to continue...

Figure 3-27 Summary of Sold, Unpaid Consignments for All Members

SUMMARY OF AMOUNT OWED TO MEMBERS FOR SOLD AND UNSOLD CONSIGNMENTS

MEMBER #	MEMBER NAME	AMOUNT SOLD	AMOUNT UNSOLD
1	DON HALEY	0.00	4.00
2	JAYNE LINK	0.00	15.00
4	HUGH KNOX	100.00	189.50
18	MARY ASH	0.00	3.00
60	PAUL WEIDHAAS	0.00	500.00
209	BECKY O'DONNELL	10.50	305.50
TOTAL AMOUNT OWED FOR SOLD AND UNSOLD CONSIGNMENT ITEMS			1127.50

Press any key to continue...

Figure 3-28 Summary of Sold and Unsold, Unpaid Consignments

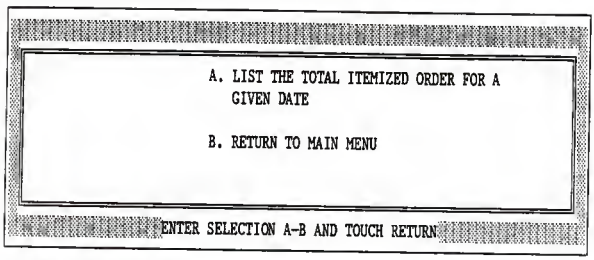

Figure 3-29 Combined Order Menu

ORDER SUMMARY FOR BLOOMING PRARIE 02/22/89

CATALOG #	DESCRIPTION	# CASES	CASE PRICE	TOTAL COST
1001	WHEAT BERRIES	1	25.00	25.00
1002	BARLEY	2	15.00	30.00
1003	CORN	2	10.00	20.00
	TOTAL ORDER COST		$	75.00

Press any key to continue...

Figure 3-30 Combined Order Report

itemized expenses for a given month and year, list gross sales and total expenses for a given month and year, and list all unpaid expenses.

List itemized expenses for a given month and year: When this action is chosen, the user is prompted to enter the month and year for which to list expenses. When the month and year have been entered, the date the expense was incurred, who was paid, a description of the goods or services, and the amount of the expense as well as the sum of the months expenses are listed (figure 3-32).

List gross sales and total expenses for a given month and year: When this action is chosen, the user is prompted to enter a month and year for which to generate a report. When the month and year have been entered, the gross sales and total expenses for a chosen month and year are listed (figure 3-33). The figures given are lump sums with no indication as to the nature of the expenses or sales.

List all unpaid expenses: When this action is chosen, all unpaid bills are listed (figures 3-34 & 3-35). Itemized in the report are who is owed, a description of the goods or services received, and the amount as well as the sum of all outstanding expenses. Additionally, consignments for which members have not been paid are summed and the total amount owed to members for

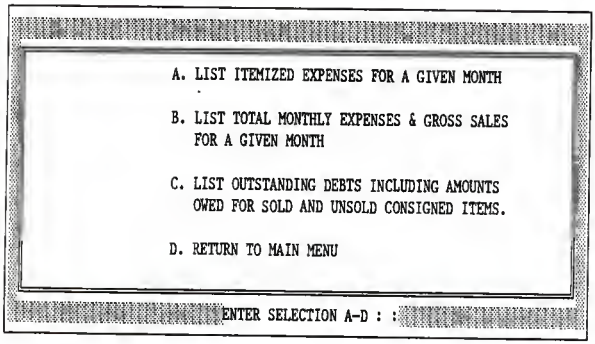

Figure 3-31 Finances Reports Menu

SUMMARY OF STORE EXPENSES FOR THE MONTH OF March 1989

DATE	PAID TO	DESCRIPTION	AMOUNT
	TOTAL MONTHLY EXPENSES		$ 0.00

Press any key to continue...

Figure 3-32 Expense Report For a Given Month and Year

MONTHLY GROSS SALES / EXPENSE REPORT

FOR THE MONTH OF March 1989

TOTAL GROSS SALES FOR THIS MONTH	$	0.00
TOTAL EXPENSES FOR THIS MONTH	$	0.00

Press any key to continue...

Figure 3-33 Report of Sales and Expenses for a Given Month and Year

REPORT OF ALL OUTSTANDING DEBTS

UNPAID BILLS

PAID TO	DESCRIPTION	AMOUNT
KPL	ELECTRICITY	20.00
	TOTAL UNPAID BILLS	$ 20.00

Press any key to continue...

Figure 3-34 Report of Unpaid Expenses

UNPAID CONSINGMENTS

NAME	CONSIGNMENT AMOUNT
DON HALEY	4.00
JAYNE LINK	15.00
HUGH KNOX	289.50
MARY ASH	3.00
PAUL WEIDHAAS	500.00
BECKY O'DONNELL	308.50
TOTAL UNPAID CONSIGNMENTS	$ 1,120.00
TOTAL OUTSTANDING DEBT	$ 1,140.00

Press any key to continue...

Figure 3-35 Summary of Outstanding Debt

consignments is given. The combined total owed for both expenses and unpaid consignments is given as well.

3.5 Summary

The system allows members of Peoples' Grocery to expedite the process of ordering goods through the grocery from the submission of an order to a supplier, through the distribution of the order to participating members. It also allows the management of the financial status of the grocery.

To take care of data entry, the system uses a series of screen forms which most people should be able to fill out with a small amount of instruction and with reduced danger of these users corrupting the database.

The reports generated by the system require a minimum of data entry by the user.

Chapter 4

Rbase 5000 Implementation

4.0 Introduction:

The RBASE implementation of the Peoples' Grocery database uses the same logical design and accomplishes the same objectives as the dBase III+ implementation though not in the same manner due to differences between the two systems. As with the dBase III+ system the Rbase system is menu driven and this chapter will examine the actions taken in response to choosing the various menu options.

Unlike the dBase III+ application which can access all tasks from the main menu, the RBASE 5000 application must access one of two submenus before any work can be initiated. These sumbenus are entitled; Enter/Update Information, and Reports (see figue 4-1). When the Enter/Update Information option is chosen, another menu (figure 4-2) is displayed which lists the areas for data entry. Choosing any of the data entry/update options displays another menu which lets the user add new information to the database or to update existing information, or to return to the data entry/update options menu.

Information is entered into the database or updated via a form tailored to each entity. When each field available in a form has been accessed, a system generated menu is displayed which allows the user to save the form, abandon it without changes to the database or to reedit the form. Saving or abandoning the form returns the user to the menu from which the form was accessed.

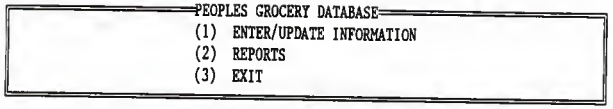

Figure 4-1 RBASE 5000 Main Menu

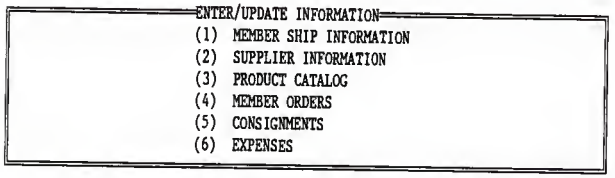

Figure 4-2 Enter/Update Options Menu

The RBASE 5000 system also generates reports of a predetermined format. When the 'Reports' option is chosen from the main menu, A menu (figure 4-3) listing report category options is displayed. As with the dBase III+ implementation, the reports generated require a minimum of user input. Though both the dBase III+ application and the RBASE 5000 application preform essentially the same the tasks, the RBASE 5000 does so with significantly less code, 1500 lines of code in 25 procedures.

4.1 Enter/Update information:

When this action is chosen, it displays a menu (figure 4-2) that allows the user to enter or update information for the following entities, Member Information, Product Catalog, Supplier Information, Member Orders, Consignments, and Expenses.

Member information:

When this option is chosen it displays a menu (figure 4-4) with the options of adding new member information or updating information for an existing member or returning to the previous menu.

Add a new member: When this option is chosen, a form is displayed (figure 4-5) allowing the user to enter a members identification number, first name, last name, street address, city, state, zipcode, home phone, work phone number. Information must be entered in the identification number and name fields, and duplicate identification numbers are checked

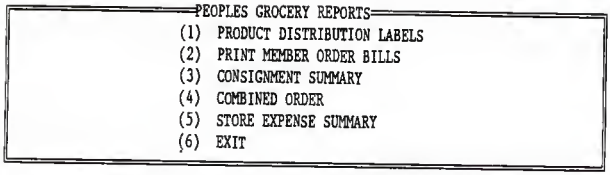

Figure 4-3 Report Options Menu

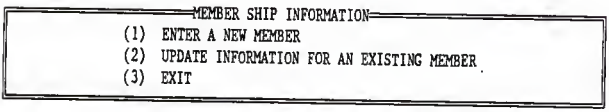

Figure 4-4 Member Information Menu

for and disallowed. When the form has been completed, it can be saved edited, or abandoned. Saving or abandonment returns the user to the member information menu.

Update an existing member: When this action is chosen, it results in a prompt asking the user to enter the identification number of the member of interest. If the number entered does not exist the user is returned to the member information menu. When a valid number has been entered, the appropriate form is displayed and the user may make changes to any field. After updating, the new information may be saved or the operation may be abandoned with no changes made to the database.

Supplier Information:

When this action is chosen, a menu giving the user the option of entering a new supplier into the database or updating an existing supplier is displayed (figure 4-6).

Enter a new supplier: When this action is chosen, a form is displayed (figure 4-7) that allows the user to enter the supplier name, street address, city, state, zipcode, contact name, and phone number.

When completed, the form may be saved, reedited, or abandoned with no changes made to the system. A supplier name must be entered and duplicate supplier names are checked. If no supplier name has been entered or a duplicate is found in the

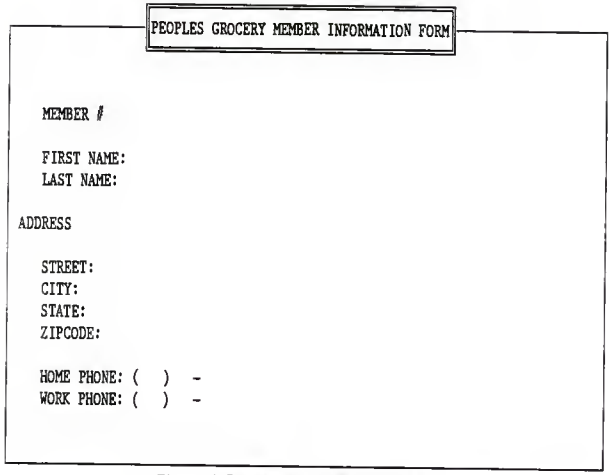

Figure 4-5 Member Information Form

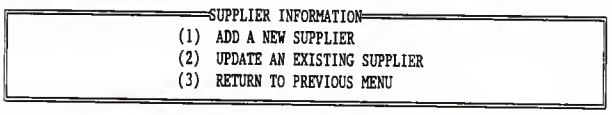

Figure 4-6 Supplier Information Menu

database, an error message is displayed and the user is given a chance to correct the error.

Update an existing supplier: When this action is chosen, the user is prompted for the name of the supplier to update. If the name entered does not exist in the database, no message is given and the user is returned to the supplier information entry/update menu. If the supplier does exist, the appropriate form is displayed and the user may change any field before saving or abandoning the operation.

Member Order:

When chosen, displays a menu (figure 4-8) allowing the user to enter new order information, or update existing information.

Enter a new order: When this action is chosen, a form is displayed (figure 4-9) that allows the user to enter the member identification number, order date, supplier name, product identification number, and the amount ordered. When the form is complete, it may be saved or the operation abandoned. Information must be entered in all fields and duplicate records are not allowed. If any field is left blank or a duplicate record is found, an error message is displayed and the user is given the opportunity to make corrections before saving or abandoning the operation.

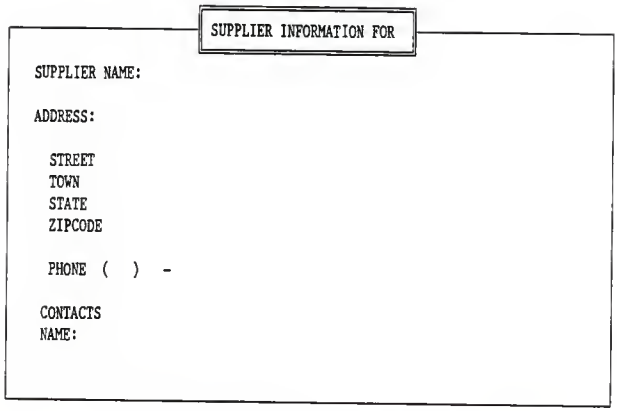

Figure 4-7 Supplier Information Form

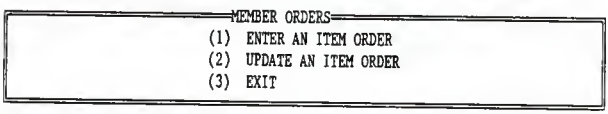

Figure 4-8 Member Order Menu

Update an existing order: When this option is chosen, the user is prompted to enter a member identification number and the date of the order. If the designated member has submitted an order on the given date, all of the items the member ordered on that date are listed and changes may be made to any field before saving or abandoning the changes. If a record matching the entered information does not exist, the user is returned to the member order entry/update menu.

Product Catalog:

When this action is chosen, it displays a menu (figure 4-10) that allows the user to add a new item to the product catalog, update existing items, or return to the enter/update menu.

Add a new item: When this action is chosen, a form is displayed (figure 4-11) that allows information for the supplier name, product identification number, product description, case price, unit price and shipping weight to be entered. After all fields have been accessed, the user may save the new record or abandon the operation. all the fields in this record must have an entry. If a field is left blank or an attempt is made to enter a duplicate record, an error message is displayed and the user is allowed to correct the error and then save the new record or abandon the operation.

Update an existing product: When this action is chosen, the

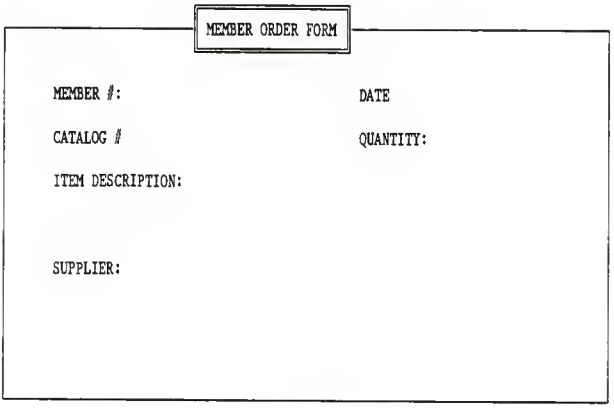

Figure 4-9 Member Order Form

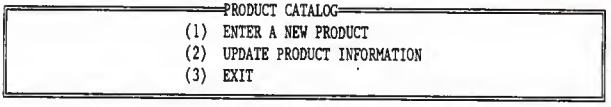

Figure 4-10 Product Information Menu

user is prompted to enter the name of a supplier and the product identification number. If a record with the requested information exists, it is retrieved and the user may modify any field before saving the new information or abandoning the operation. If a matching record is not found, the user is returned to the update menu.

Consignments:

When this option is chosen a menu is displayed (figure 4-12) that allows the user to enter a new consignment, update existing consignments or return to the enter/update menu.

Enter a new consignment: When this action is chosen, a consignment form is displayed (figure 4-13) in which the user enters the member identification number, date, the consignment description, the number of the given item placed on consignment, the price to be paid to the consignor for each item, whether or not all items have been sold, and whether or not the member has been paid for the consignments sold. Entries must be made in the member identification, description, date, amount paid and number of items fields. Duplicate records are not allowed. The existence of a duplicate record or the failure to enter data in a field that requires data to be entered

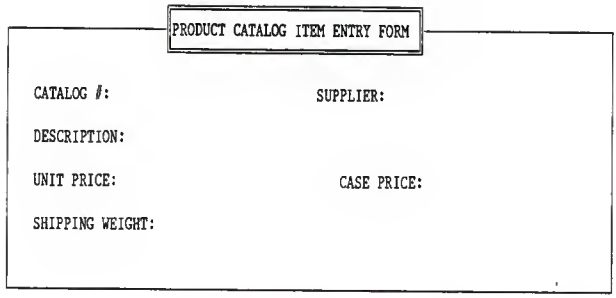

Figure 4-11 Product Information Form

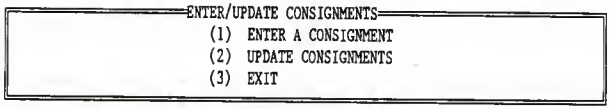

Figure 4-12 Consignment Menu

prompts an error message and the user is allowed to make the necessary corrections before saving the record or abandoning the operation.

Update an existing consignment: When this action is chosen, the user is prompted to enter the identification number of the member whose consignments they want to update. All existing consignments for this member are listed and the user may make changes to any field of any record and then save or abandon the modifications. If the given member has no consignments on record, the user is returned immediately to the consignment entry/update menu.

Expenses:

When this action is chosen, it displays a menu (figure 4-14) that allows the user to enter a new expense or to update an existing expense.

Enter a new expense: When this action is chosen, a form is displayed (figure 4-15) that allows the user to enter the date, who was paid, a description of the goods or services received, the amount of the expense and whether or not the expense has been paid. All fields but the expense paid field must have an entry and as usual, duplicate records are not allowed. If any field but the 'expense paid' field is left empty or a duplicate record is found, an error message is displayed and the user is

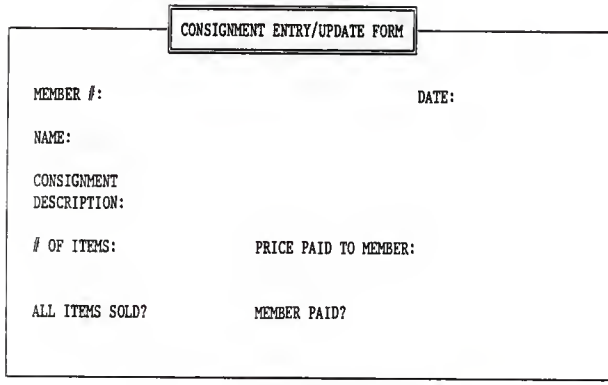

Figure 4-13 Consignment Information Form

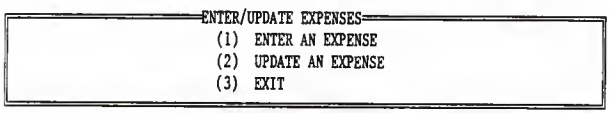

Figure 4-14 Expense Information Menu

given the opportunity to make necessary corrections after which the user may save the record or abandon the operation.

Update existing expense records: When this action is chosen, all existing expense records are displayed and the user may update any field in any record after which changes may be saved or the update operation abandoned.

4.2 REPORTS

When the report option is chosen from the main menu, the menu (figure 4-3) listing the report categories is displayed. These categories are product breakdown labels, member order bills, combined order summary, consignments, and finances. This section will describe the actions taken when these options are chosen.

Product breakdown labels:

When this action is chosen, another menu (figure 4-16) is displayed that allows the users to print a product breakdown label for a selected order or to return to the main menu.

Print product breakdown label for a specified order: When this action is chosen, the user is prompted to enter an order date and the name of the supplier ordered from. If no such date / supplier combination exists in the database, all order dates and suppliers are listed and the user is allowed to reenter the requested information. Upon entry of a valid date and supplier

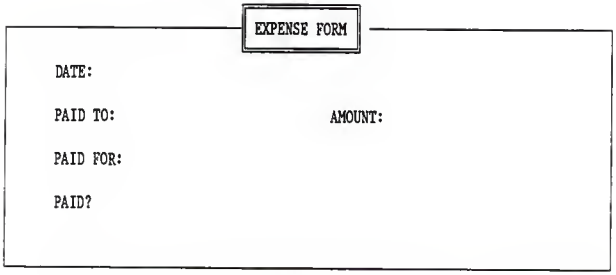

Figure 4-15 Expense Data Form

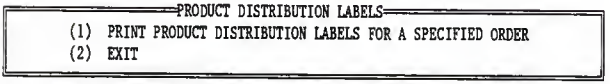

Figure 4-16 Product Breakdown Label Menu

a product breakdown label for each item ordered is printed (figure 4-17). Listed are the item name, the names of the members ordering the item and the amount of the item each member ordered.

Member order bills:

When this action is chosen, another menu (figure 4-18) is displayed that allows the users to print an itemized bill for each member participating in a selected order or return to the main menu.

Print bills for members participating in a selected order: When this action is chosen, the user is prompted to enter an order date and the name of the supplier ordered from. If no such date / supplier combination exists in the database, all order dates and suppliers are listed and the user is allowed to reenter the requested information. Upon entry of a valid date and supplier the user may change information that varies from order to order. The items that may be changed at this point are the tax rate, percentage of markup, shipping cost, member equity charges, and discount rate. After these changes have been made the user is prompted to enter any items that were not delivered because they were out of stock, damaged in transit or otherwise unacceptable. These products are deleted from the order and a bill is printed for each member participating in the order (figure 4-19).

ITEM DISTRIBUTION LABEL

Item Description: CHICKEN

MEMBER NAME	QUANTITY
BECKY ODONNELL	5.

Figure 4-17 Product Breakdown Label Report

ORDER BILLS

(1) PRINT ITEMIZED BILLS FOR EACH MEMBER IN A SPECIFIED ORDER
(2) EXIT

Figure 4-18 Order Bill Menu

Combined order summary:

When this action is chosen, another menu (figure 4-20) is displayed that allows the users to print a summary of the items ordered for a selected order or to return to the main menu.

Print an order summary for a specified order: When this action is chosen, the user is prompted to enter an order date and the name of the supplier ordered from. If no such date / supplier combination exists in the database, all order dates and suppliers are listed and the user is allowed to reenter the requested information. Upon entry of a valid date and supplier a summary of the order is printed (figure 4-21). Listed are the product identification number, the item description, the number of cases ordered, the total value of the ordered item and the value of the whole order.

Consignments:

When this action is chosen, another menu (figure 4-22) is displayed which lists the reports that are produced by this section. These reports are list products sold but member unpaid for a given member, list total items sold but members unpaid for all members, and list total amount owed for sold and unsold consignments.

List products sold but member unpaid for a given member: When this action is chosen, the user is prompted to enter the desired members identification number. This member's

MEMBER ORDER BILL

Member Name: BECKY ODONNELL

CATALOG #	OESCRIPTION	QUANTITY	PRICE
201	CHICKEN	5.	$10.00
	VALUE OF ORDER		$10.00
	MEMBER EQUITY CHARGE		$0.20
	MARKUP PRE EQUITY TOTAL X 15. percent		$1.50
	TAX 4.5 percent		$0.52
	SHIPPING COST		$0.40
	TOTAL CHARGE		$12.62

Figure 4-19 Order Bill Report

COMBINED ORDER REPORT

(1) ITEMIZE TOTAL ORDER FOR A GIVEN DATE ANO SUPPLER
(2) EXIT

Figure 4-20 Order Summary Menu

ORDER SUMMARY

DATE: 11/21/88 SUPPLER: blooming prarie

ITEM#	DESCRIPTION	TOTAL QUANTITY	VALUE
201	CHICKEN	5.	$10.00
	TOTAL VALUE OF ORDER		$10.00

Figure 4-21 Order Summary Report

CONSIGNMENT SUMMARY

(1) LIST PRODUCTS SOLD BUT MEMBER UNPAID FOR A GIVEN MEMBER
(2) LIST TOTAL ITEMS SOLD BUT MEMBERS UNPAID FOR ALL MEMBERS
(3) LIST TOTAL AMOUNT OWED FOR SOLD AND UNSOLD CONSIGNMENTS
(4) EXIT

Figure 4-22 Consignment Reports Options Menu

consignments which have been sold but for which the member has not been paid are then itemized ant the total amount owed the member is displayed (figure 4-23). If the chosen member has no sold, unpaid consignments, a list of all members with sold unpaid consignments is displayed and the user is allowed to reenter a new identification number.

List amount for items sold but members unpaid for all members: When this action is chosen, a form is produced (figure 4-24) which lists each member who has consignments which have been sold but for which the member has not been paid. Also given is the amount owed to each of these members and the total amount owed to all members.

Expenses:

When this option is chosen, a menu is displayed (figure 4-25) that lists three possible expense reports. These reports are 1) itemize expenses for a given month, 2) total expenses and gross sales for a given month, and 3) total outstanding debt.

Itemize expenses for a given month: When this option is chosen, the user is prompted to enter a date for which to itemize expenses. Upon entry of the chosen date, a form (figure 4-26) listing the description of the expense, the amounts paid, who was paid and the total expenses for the month is produced.

SUMMARY OF SOLD BUT UNPAID CONSIGNMENTS
Member Name: BECKY ODONNELL

DESCRIPTION	UNITS	UNIT PRICE	VALUE
eggs	5	$0.75	$3.75
tomatoes	5	$0.25	$1.25
FLOUR	1	$1.00	$1.00
TOTAL SOLD BUT UNPAID CONSIGNMENTS			$6.00

Figure 4-23 Consignments Sold but Member Unpaid for a Given Member

SUMMARY OF SOLD UNPAID CONSIGNMENTS

MEMBER NAME	AMOUNT
BECKY ODONNELL	$1.25
TOTAL SOLD BUT UNPAID CONSIGNMENTS	$1.25

Figure 4-24 Summary of Sold Unpaid Consignments for All Members

SUMMARY OF UNPAID CONSIGNMENTS SOLD AND UNSOLD

MEMBER NAME	AMOUNT
JOE SMITH	$9.00
BECKY ODONNELL	$12.25
SUMMARY OF TOTAL UNPAID CONSIGNMENTS	$21.25

Figure 4-25 Sold and Unsold Consignments Report

```
                    EXPENSE REPORT MENU
    (1)  LIST ITEMIZED EXPENSES FOR A GIVEN MONTH
    (2)  LIST TOTAL EXPENSES AND GROSS SALES FOR A GIVEN MONTH
    (3)  LIST OUTSTANDING DEBTS INCLUDING AMOUNT OWED FOR CONSIGNMENT
    (4)  EXIT
```

Figure 4-26 Expense Report Options Menu

Total expenses and gross sales: When this action is chosen, the user is prompted to enter the date for which to produce the report. Upon entry of a date, a form is produced (figure 4-27) which lists the total expenses of the grocery for the designated month along with the total gross sales of the grocery for the same month.

Outstanding debt: when this action is chosen a form is produced (figure 4-28) which lists the total amount owed for unpaid expenses and the total amount owed for consignments.

Summary:

The RBASE 5000 application performs the same tasks as the dBase III+ application. However, there are some system dependent differences in accomplishing these tasks. The next chapter will qualitatively compare RBASE 5000 and dBASE III+.

SUMMARY OF ALL EXPENSES FOR MONTH OF November 1988

DESCRIPTION	AMOUNT
ART WORK	$5.00
TYPING	$5.00
painting	$8.00
TOTAL EXPENSES FOR THE MONTH	$18.00

Figure 4-27 Itemized Expense Report for a Given Month

SALES / EXPENSE REPORT FOR THE MONTH OF November 1988

TOTAL VALUE OF GOODS ORDERED FROM SUPPLIERS THIS MONTH	$10.00
TOTAL MONTHLY EXPENSES	$0.00

Figure 4-28 Expense and Gross Sales Report for a Given Month

Chapter 5

Comparison

5.0 Introduction:

Given the previously described enterprise, and implementations of dBase III+ and RBASE 5000, several comparisons can be made which can help judge the suitability of each system for use in developing other prospective database implementations. Among the possible points of comparison are 1) the "user friendliness" of the system, 2) ease of implementation, 3) the ease with which the system may be adapted to a specific application.

5.1 User friendliness:

The ability to develop a user friendly system is a primary concern when building a front-end for a database management system. In the case of the Peoples' Grocery database, the members using the system will change frequently and they will come to use the system with a widely varying knowledge of how the database is organized or even how to use a computer. Therefore, it is essential that the front-end lead the user through the necessary operations while allowing them to easily make corrections or return to the starting point without changing the database.

One of the important attributes of a "user friendly" system is the consistency of command actions [SIME 85] that is a response to a command, no matter how often or where the command appears, should

result in the same action. The command structure of an application developed with RBASE 5000 is dictated primarily by the code generation facility which enforces consistency of command actions. The application generation facility creates a tree of menus and actions taken as options of these menus. This tree must be traversed menu by menu. Likewise when the action or the use of a menu is finished, the user is returned to the previous menu.

The command structure of dBASE III+ gives much more flexibility to the programmer. Menus and the actions taken from those menus are determined solely by the programmer writing the application. This could lead to a breakdown in the consistency of program behavior but it is a simple matter to develop a pattern of menu and user query presentation which can be maintained throughout an application.

Screen formatting is an important aspect of user friendliness and again RBASE 5000 exerts more rigid control over final screen appearance than dBASE III+. When designing menus in RBASE 5000 the user enters the menu title and option headings and the system automatically formats the screen using the information given (figure 4-x). Slightly more flexibility is given when developing forms or reports in that the user may place text wherever desired and place form borders wherever wanted. But, there is no choice of what may used for the border or the option of using other visual amenities to make the screen more attractive.

DBASE III+, to the contrary, gives the programmer full control over how the screen will appear. The programmer can determine the dimensions of a menu or form and what to use as the border (figure

3-1). This flexibility of course places a greater burden on the programmer but can produce a more varied and interesting result.

5.2 Implementation considerations:

From the implementation standpoint, comparisons may be made between dBASE III+ and RBASE 5000 for the following points: 1) the code generation capabilities of each system, 2) the size of the code needed to accomplish the same work on each system, 3) the implementation of the rules used to select records and maintain the integrity of the database.

Code generation:

Both dBASE III+ and RBASE 5000 are able to generate code for use in specific applications but there are significant differences in the complexity of the code each can generate. The code generated by dBASE III+ is rudimentary and impractical to use.

This token gesture at code generation is in sharp contrast to the way code is produced in RBASE 5000. Code generation in RBASE 5000 is integral to the system. In fact, little useful work can be performed, from a program, without using the code generation facilities. When used, the code generator leads the user through a logical series of steps to build menus, and define the actions that may be taken by choosing options contained in the menus. Menus may be nested up to seven levels deep. Data is entered into the database via predefined forms and user produced code can be incorporated

where desired. In this manner, with a little experimentation, simple applications can be developed fairly quickly and the person preparing the code is rewarded with the sense of having accomplished a significant amount of work in a relatively short amount of time.

Overall, the code generation facility of RBASE 5000 is very well done but there are at least two shortcomings worth mentioning. First, the application generator keeps track of the names of the subroutines that the user has incorporated, and it prevents the programmer from using a given subroutine more than once. Thus, frequently used code that would normally be placed in a subroutine and called repeatedly from various points in a program must be placed wherever it is needed, as often as it is needed.

The second deficiency concerns code modification. The user generated pieces of code are developed via the Rbase 5000 editor. However, once a given piece of code has been incorporated into an application via the code generation facility, that code must thence forth be modified from the code generation facility. This requirement is not clearly stated in the manuals provided and it necessitates a considerable amount of effort since the application generation procedure must be traversed to the point of modification,the modification made, and the application recompiled whenever a change is desired.

Size of code:

In the Peoples' Grocery implementations, the RBASE 5000 version occupied 41000 bytes while the dBASE III+ version used 81000.

The code used by RBASE 5000 was gathered into three RBASE files which contain the database records and associated information and the various application files produced by the user and the code generator during application implementation.

The dBASE III+ implementation was not nearly as tidy. Unlike RBASE 5000, dBASE III+ does not permit several subroutines to be stored in a single file and the dBASE III+ editor can only handle files with a maximum size of 5000 bytes. Consequently, the dBASE III+ implementation of the Peoples' Grocery database is spread out over 99 program files plus the data files.

Maintaining system integrity:

Both RBASE 5000 and dBASE III+ provide methods for filtering data before it is committed to the database but their methods differ markedly. To assure database integrity, RBASE 5000 maintains a table of rules that is built as an application is being developed. Whenever the situation arises that requires the validity of a piece of data to be checked before it is allowed into the database, a condition against which the data must be compared is constructed and added to the rule table. When the rule checking is activated, all fields in a record are checked to see that they meet the constraints of every pertinent rule existing in the rule table before the record is written to the database. This behavior is too restrictive.

To maintain data integrity in dBASE III+, the user establishes the conditions to be checked and compares incoming data against those conditions at the appropriate place in the program. A given

set of conditions are used only where they are applicable.

Summary:

With the user given virtually total control of all aspects of dBASE III+ application implementations, the initial development of an application proceeds slowly but improves as the user gains experience., The systems integrity can be strictly maintained and a consistent, easy to use system interface can be developed which will provide for modifications as the need arises.

In the RBASE 5000 environment, the broad aspects of application development are controlled by the system thus the data entry and update tasks as well as the outline of other portions of the system can be developed very quickly. Data manipulation and report generation proceed more slowly until considerable experience has been gained using the programming commands provided by the system. The final product is predictable in its reaction to user responses to application prompts.

Table 5-1 summarizes the comparison of the friendliness of the user interface and the facilities available for implementation of applications in both dBASE III+ and RBASE 5000. Chapter 6 will discuss these points further.

	RBASE 5000	dBase III+
Command Actions	System enforced; consistent	User defined; potentially inconsistent
Screen Formatting	Menus; dictated by system	All aspects under user control
	Forms and Reports; User defined	
Code Generation Capabilities	Integral to system	Rudimentary
Size of Code	Relatively compact; 1500 lines of code in 25 procedures	Large; 2500 lines of code in 99 procedures
Integrity Maintenence	Rigid; all or none	Flexible; may be enforced only at point of data entry

Figure 5-1 System Comparison Table

Chapter 6

Conclusion

Both dBASE III+ and RBASE 5000 proved adequate for developing a front-end application for a small enterprise. Each system excelled in certain areas and each also had deficiencies that are worthy of further discussion. This chapter will review both the areas of excellence and the shortcomings of both systems.

To its credit RBASE 5000 has an excellent code generation facility that allows the programmer to produce the bulk of an application quickly. Once the menu and data entry facilities are in place, the programmer can incorporate hand written code specifically designed for the application. Navigation through applications thus produced is consistent and easy to learn. The code data for an application can be kept in a few files which are easy for a database administrator to maintain. RBASE 5000 has an excellent on line help facility. This facility is needed when writing code because the syntax of the programming commands is clumsy and unforgiving of misplaced spaces and the error messages produced when a syntax error is encountered are not very helpful in identifying and rectifying the problem. Another and perhaps the most serious drawback of RBASE 5000 is the interruption of and potential termination of program execution if the keyboard is touched when the application is not expecting keyboard input. This behavior is very disconcerting and detrimental to the overall friendliness of the system.

There are no system produced, surprise exits in dBASE III+. The

programming commands are plentiful and their use, with a little practice is intuitive. Though a significant burden is placed on the programmer for the development of an application, "boiler plate" programs simplify development significantly and accomplish in essence, the service the RBASE 5000 code generator provides.

At this point one application has been implemented using two microcomputer based database management system products. This leaves many products as well as many application environments unexplored. It would prove useful to select a set of application environments with a variety of requirements with regards to security, data integrity, flexibility of the user interface, and size of the database to name a few. Once a cross section of application environments has been chosen, each could be implemented using the existing database management system products. This would provide a bench mark which potential users might consult when considering acquisition of a DBMS. The user could choose the bench mark environment that most closely matches their requirements and evaluate the relative strengths and weaknesses of a DBMS based on how it preforms in that environment.

As new DBMS software becomes available in the market place, and existing DBMS are updated, it would prove useful to continue to qualitatively compare them using practical, real life business application bench marks in addition to the technical quantitative tabular comparisons commonly employed in software evaluation.

BIBLIOGRAPHY

[Asht 86] DBASE III+ Version 1.0 Manual. Ashton Tate 1986.

[Bern 82] Bernstein, P.A. Notes on Database Management Systems. Aiken Computation Laboratory, Harvard Univ. Cambridge Mass. 1982.

[Dick 86] Dickinson, J. "Programmable Relational Databases" PC Magazine 5:12 (1986).

[Eric B6] Erickson, J. and Baran, N. Using RBASE 5000. Osborne Mcgraw-Hill 1986.

[Gabe 86] Gabel D. "Designing a Database Application" Supplement to PC Week 3:17 (1986)

[Kroe 83] Kroenke, D. Database Processing. Science Research Associates 1983.

[Micr 86] R:BASE System V Learning Guide. Version 1.0. Microrim 1986.

[Shac 86] Shackel B. "Ergonomics in Design for Usability." In 'People and Computers: Designing for Usability. Proceedings of the Second Conference of the British Computer Society Human Computer Interaction Specialist Group.' Univ. of York 23-26 Sept. 1986 M.D. Harrison, A.F. Monk (eds.). pp. 44-77 Cambridge University Press, Cambridge.

[Sime 85] Simes, D.K. and Sirsky, P.A. "Human Factors: An Exploration of the Psychology of Human Computer Dialogues." In 'Advances in Human-Computer Interaction.' vol 1, Hartson, H.R. ed. Ablex Publishing Corp., Norwood, New Jersey. 1985

[Ullm 82] Ullman, J.D. Principles of Database Systems. Computer Science Press 1982.

```
*PROGRAM PEOPLESM
*WRITTEN FOR PEOPLES GROCRERY DATABASE
*BY GARY RADKE
*MAIN ROUTINE

CLEAR ALL
SET SCOREBOARD OFF
SET TALK OFF
SET ESCAPE OFF
SET BELL OFF
SET HEADING OFF
SET HELP OFF
SET MENU OFF
SET SAFETY OFF
SET STATUS OFF

DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO MenuTEMP
        DO WHILE .T.
          i=INKEY()
          @ 15,58 SAY ""
          IF UPPER(CHR(i))$"ABCDEFGHIJKX"
             EXIT
          ENDIF
          i=0
          @ 15,58 SAY UPPER(CHR(i))
        ENDDO

   * process user's response
   DO CASE
      CASE CHR(i) $ "Aa"
         DO MEMINF

      CASE CHR(i) $ "Bb"
         DO MEMORD1

      CASE CHR(i) $ "Cc"
         DO NEWCON1

      CASE CHR(i) $ "Dd"
         DO SUPP

      CASE CHR(i) $ "Ee"
         DO PRODUCT
```

```
        CASE CHR(i) $ "Ff"
            DO EXPENSES


        CASE CHR(i) $ "Gg"
            DO LABELS

        CASE CHR(i) $ "Hh"
            DO BILLING

        CASE CHR(i) $ "Ii"
            DO CONSSUMM

        CASE CHR(i) $"Jj"
            DO STOREORD

        CASE CHR(i) $ "Kk"
            DO FINANCES

        CASE CHR(i) $ "Xx"
            EXIT
    ENDCASE

ENDDO
SET TALK ON
SET ESCAPE ON
SET BELL ON
SET HEADING ON
SET HELP ON
SET MENU ON
SET SAFETY ON
SET STATUS ON
CLEAR ALL
CLEAR
RETURN
* Eof:


* Program.: MENUTEMP.PRG
* Author..: Gary Radke
* Date....: March 1987
* Notes...: Menu screen for Peoples Grocery database

@ 1,21 TO 3,51
@ 4,1 TO 23,77 DOUBLE
@ 6,3 TO 14,37
@ 5,4 TO 7,21 DOUBLE
@ 6,5 SAY SPACE(16)
@ 6,41 TO 14,75
@ 5,42 TO 7,53 DOUBLE
@ 6,43 SAY SPACE(10)
```

```
@ 5,2 SAY CHR(176)+CHR(176)
@ 6,2 SAY CHR(176)
@ 7,2 SAY CHR(176)
@ 8,2 SAY CHR(176)
@ 9,2 SAY CHR(176)
@ 10,2 SAY CHR(176)
@ 11,2 SAY CHR(176)
@ 12,2 SAY CHR(176)
@ 13,2 SAY CHR(176)
@ 14,2 SAY CHR(176)
@ 16,2 SAY REPLICATE(CHR(176),75)
@ 17,2 SAY REPLICATE(CHR(176),75)
@ 18,2 SAY REPLICATE(CHR(176),75)
@ 19,2 SAY REPLICATE(CHR(176),75)
@ 20,2 SAY REPLICATE(CHR(176),75)
@ 21,2 SAY REPLICATE(CHR(176),75)
@ 22,2 SAY REPLICATE(CHR(176),75)
@ 14,76 SAY CHR(176)
@ 13,76 SAY CHR(176)
@ 12,76 SAY CHR(176)
@ 11,76 SAY CHR(176)
@ 10,76 SAY CHR(176)
@ 9,76 SAY CHR(176)
@ 8,76 SAY CHR(176)
@ 7,76 SAY CHR(176)
@ 6,76 SAY CHR(176)
@ 5,54 SAY REPLICATE(CHR(176),23)
@ 5,22 SAY REPLICATE(CHR(176),19)
@ 6,38 SAY REPLICATE(CHR(176),3)
@ 7,38 SAY REPLICATE(CHR(176),3)
@ 8,38 SAY REPLICATE(CHR(176),3)
@ 9,38 SAY REPLICATE(CHR(176),3)
@ 10,38 SAY REPLICATE(CHR(176),3)
@ 11,38 SAY REPLICATE(CHR(176),3)
@ 12,38 SAY REPLICATE(CHR(176),3)
@ 13,38 SAY REPLICATE(CHR(176),3)
@ 14,38 SAY REPLICATE(CHR(176),3)
@ 2,24 SAY "PEOPLES GROCERY DATABASE"
@ 6,6 SAY "INPUT / UPDATE"
@ 6,44 SAY "REPORTS"
@ 8,8 SAY "A. Member Info."
@ 9,8 SAY "B. Member Orders"
@ 10,8 SAY "C. New Consignments"
@ 11,8 SAY "D. Suppliers"
@ 12,8 SAY "E. Product Catalogs"
@ 13,8 SAY "F. Store Expenses"
@ 8,46 SAY "G. Product / Member Labels"
@ 9,46 SAY "H. Members Order Bill"
@ 10,46 SAY "I. Consingment Summary"
@ 11,46 SAY "J. Combined Order"
@ 12,46 SAY "K. Finances"
@ 15,18 SAY "[Enter Selection (A - L, or X to quit) : :]"
```

```
@ 15,2 SAY REPLICATE(CHR(176),16)
@ 15,61 SAY REPLICATE(CHR(176),16)
RETURN
* Eof:Menutemp.prg


*PROGRAM MEMINF.PRG
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE

CLEAR
SELECT 1
USE MEMBER INDEX MEMNO
* Cursor coordinates in menu queries
XIT=.F.
ERR=.T.
XCORD=11
YCORD=51
* Messages and menu queries
ERRMESS=SPACE(60)
DUPMESS="THIS MEMBER # IS ALREADY IN USE, PLEASE USE ANOTHER."
DUPNUM="MEMBER # "
MEMESS="UPDATE MEMBERSHIP INFO FOR THIS MEMBER? Y/N"
YMESS="Y - UPDATE INFORMATION FOR THIS MEMBER."
NMESS="N - SKIP THIS MEMBER, LOOK AT THE NEXT."
XMESS="X - RETURN TO THE PREVIOUS MENU."
MESS1="MEMBER # IS INVALID USE ONE LESS THAN 1000"
MESS2="YOU MUST ENTER A MEMBER NUMBER"
MESS3="YOU MUST ENTER A MEMBER NAME"

DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   DO MOPT

   * Accept only allowed responses
   DO CONFIRMM

   * process user's response
   MMEMNO = 0
   MLAST=SPACE(15)
   MFIRST=SPACE(9)
   MSTREET=SPACE(20)
   MCITY=SPACE(15)
   MZIP=00000
   MHOME=SPACE(13)
   MWORK=SPACE(13)
   MSTATE=SPACE(2)
```

```
    DO CASE
    * Enter a record for a new member
     CASE CHR(i) $ "Aa"
       DO NEWMEM

      * Allow update of an existing membership record
      * Excluding update of member #
     CASE CHR(i) $ "Bb"
       DO UPDTMEM

      * Return to main menu
     CASE CHR(i) $ "Cc"
       EXIT
     ENDCASE

   ENDDO
   RELEASE ALL
   CLOSE ALL
   RETURN
* Eof:

* PROGRAM MEMORD.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ENTERS AND UPDATES MEMBER PRODUCT ORDERS
CLEAR
ERRMESS=SPACE(60)
DUPMESS="THIS ORDER ALREADY EXISTS PLEASE REENTER"
MESS1="CATALOG # DOES NOT EXIST FOR THIS SUPPLIER, TRY ANOTHER"
QUANTM="YOU MUST ENTER A VALUE FOR THE QUANTITY"
XCORD=11
YCORD=51
QUERY1="UPDATE ORDER FOR MEMBER # "
QUERY2="CONTINUE ORDERING WITH A NEW MEMBER? Y/N"
QUERY3="CONTINUE ORDERING WITH A NEW SUPPLIER? Y/N"
QUERY4="UPDATE ORDER FOR MEMBER # "
SUPMESS="SUBMIT ORDER TO THIS SUPPLIER?"
SYMESS="Y - ORDER FROM THIS SUPPLIER"
SNMESS="N - DO NOT ORDER FROM THIS SUPPLIER"
SXMESS="X - RETURN TO PREVIOUS MENU"
MEMESS="SUBMIT ORDER FOR THIS MEMBER?"
YMESS="Y - SUBMIT ORDER FOR THIS MEMBER"
NMESS="N - DISPLAY NEXT MEMBER"
XMESS="X - RETURN TO PREVIOUS MENU"
MEMNAME=SPACE(20)
MSUPNAME=SPACE(20)
MMEMNO=0
MITEMNO=0
MQUANTITY=0
MDATE=DATE()
MDESCRIPT=SPACE(60)
LASTITEM=SPACE(50)
```

```
XIT=.F.
ERR=.T.
SELECT 1
USE MEMBER INDEX MEMNO
SELECT 2
USE PRODCAT INDEX NAMEITNO
SELECT 3
USE ITEM_ORD
SELECT 4
USE SUPPLIER
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   DO ORDOPT
   DO CONFIRMM

   * process user's response
   DO CASE

    * Submit a new order
      CASE CHR(i) $ "Aa"
       DO NEWORD
    * Update an existing order
      CASE CHR(i) $ "Bb"
       DO UPDTORD
    * Return to the main menu
      CASE CHR(i) $ "Cc"
       EXIT

   ENDCASE
ENDDO
CLEAR ALL
RELEASE ALL
RETURN
* Eof:

*PROGRAM CONSIGN
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
CLEAR
ERROR=.F.
XCORD=11
YCORD=51
MEMESS="UPDATE CONSIGNMENT FOR THIS MEMBER? Y/N"
YMESS="Y - DISPLAYS CONSIGNMENTS FOR THIS MEMBER."
NMESS="N - SHOWS YOU THE NEXT MEMBER."
XMESS="X - RETURNS YOU TO THE PREVIOUS MENU"
MESS1="YOU MUST ENTER A CONSIGNMENT DESCRIPTION"
```

```
MESS2="YOU MUST ENTER A VALUE FOR QUANTITY"
MESS3="YOU MUST ENTER A CONSIGNMENT PRICE"
MESS4="MEMBER # DOES NOT EXIST IN THE DATABASE"
MESS5="THIS MEMBER HAS NOTHING ON CONSIGNMENT"
MESS6="THIS CONSIGNMENT ALREADY EXISTS IN THE DATABASE"
ERRORMESS=SPACE(60)
SELECT 1
USE MEMBER INDEX MEMNO
SELECT 2
USE MEMBER_C INDEX CMEMNO
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   * print menu background
   DO CONOPT
   * print consignment options

   * accept users response
   DO CONFIRMM
   MMEMNO=0
   MEMNAME=SPACE(40)
   MDESCRIPT=SPACE(20)
   MNUMITEM=0
   MPRICE=000.00
   MSOLD=.F.
   MPAID=.F.
   MDATE=DATE()
   * process user's response
   DO CASE

      * add a consignment
      CASE CHR(i) $ "Aa"
         DO ECON

      CASE CHR(i) $ "Bb"
         DO CCON

      CASE CHR(i) $ "Cc"
         EXIT
    ENDCASE
ENDDO
CLOSE ALL
RELEASE ALL
RETURN
* Eof:
```

```
*PROGRAM SUPP
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*ENTERS SUPPLIER INFORMATION INTO SUPPLIER.DBF

CLEAR
XIT=.F.
ERR=.T.
XCORD=11
YCORD=51
MESS1="YOU MUST ENTER A SUPPLIER NAME"
MESS2="THIS SUPPLIER NAME ALREADY EXISTS IN THE DATABASE"
SELECT 1
USE SUPPLIER
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   DO SUPPOPT
   MSUPNAME=SPACE(20)
   MCONTACT=SPACE(20)
   MSTREET=SPACE(20)
   MCITY=SPACE(15)
   MZIP=00000
   MPHONE=SPACE(13)
   MSTATE=SPACE(2)
   DO CONFIRMM

   * process user's response
   DO CASE

     CASE CHR(i) $ "Aa"
       DO WHILE .T.
       DO SUPFMASK
       DO ADDSUP
       LOCATE FOR RTRIM(MSUPNAME)=RTRIM(SUPNAME)
       IF FOUND()
           @ 22,5 SAY MESS2
           WAIT
           @ 22,5
           @ 23,5
           LOOP
        ENDIF
        IF (LEN(RTRIM(MSUPNAME))<1)
           @ 22,5 SAY MESS1
           WAIT
           @ 22,5
```

```
      @ 23,5
      LOOP
   ENDIF
   DO CONFIRMQ
   DO CONFIRML

   DO CASE
      CASE CHR(i) $ "Yy"
        APPEND BLANK
        REPLACE SUPNAME WITH MSUPNAME
        REPLACE CONTACT WITH MCONTACT
        REPLACE STREET WITH MSTREET
        REPLACE TOWN WITH MCITY
        REPLACE ZIPCODE WITH MZIP
        REPLACE STATE WITH MSTATE
        REPLACE PHONE WITH MPHONE
        @ 22,5
        @ 23,5
        WAIT
        EXIT
      CASE CHR(i) $ "Nn"
        LOOP
      CASE CHR(i) $ "Xx"
        EXIT
   ENDCASE
ENDDO


* Update an existing supplier
CASE CHR(i) $ "Bb"
  CLEAR
  * Start at the top of the file and work through
  * it until the desired supplier is found
  USE SUPPLIER
  DO WHILE .T.
    MSUPNAME= RTRIM (SUPNAME)
    @ 3,5 SAY "SUPPLIER"
    @ 3,15 SAY MSUPNAME
    @ 5,5 SAY "UPDATE INFO FOR THIS SUPPLIER? Y/N"
    @ 7,0 TO 13,55 DOUBLE
    @ 8,5 SAY "Y DISPLAYS INFO FOR THIS SUPPLIER"
    @ 10,5 SAY "N SHOWS YOU THE NEXT SUPPLIER"
    @ 12,5 SAY "X RETURNS YOU TO THE PREVIOUS MENU"
    DO CONFIRML
    DO CASE
      CASE CHR(i) $ "Yy"
        MSUPNAME=SUPNAME
        EXIT
      CASE CHR(i) $ "Nn"
        SKIP
        IF EOF()
          @ 22,5 SAY "END OF SUPPLIER FILE"
```

```
                WAIT
                @ 22,5
                @ 23,5
                EXIT
              ELSE
                @ 3,15
              ENDIF
            CASE CHR(i) $  "Xx"
              EXIT
          ENDCASE
        ENDDO

        IF (CHR(i) $ "Xx") .OR. (EOF())
          LOOP
        ENDIF
        * Update the chosen supplier record
        MCONTACT = CONTACT
        MSTREET = STREET
        MCITY = TOWN
        MZIP = ZIPCODE
        MSTATE = STATE
        MPHONE = PHONE
        DO WHILE .T.
           DO SUPFMASK
           DO UPSUPRD
           @ 22,5 SAY "IS THIS INFORMATION CORRECT? Y/N"
           @ 23,5 SAY "TYPE X TO RETURN TO THE PREVIOUS MENU"
           DO CONFIRML
           DO CASE
              CASE CHR(i) $ "Yy"
                REPLACE CONTACT WITH MCONTACT
                REPLACE STREET WITH MSTREET
                REPLACE TOWN WITH MCITY
                REPLACE ZIPCODE WITH MZIP
                REPLACE STATE WITH MSTATE
                REPLACE PHONE WITH MPHONE
                @ 22,0
                @ 23,0
                WAIT
                EXIT
              CASE CHR(i) $ "Nn"
                LOOP
              CASE CHR (i) $ "Xx"
                EXIT
           ENDCASE
        ENDDO

     * Return to main menu
     CASE CHR(i) $ "Cc"
        EXIT
   ENDCASE
ENDDO
```

```
CLEAR ALL
RELEASE ALL
RETURN
* Eof:

* PROGRAM PRODUCT
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ENTER AND UPDATE ITEMS IN A SUPPLIERS CATALOG

CLEAR
SELECT 1
USE SUPPLIER
SELECT 2
USE PRODCAT INDEX PRODORD
XCORD=11
YCORD=51
ERROR=.T.
XIT=.F.
ERRMESS=SPACE(60)
MESS1="THIS CATALOG # ALREADY EXISTS FOR THIS SUPPLIER, REENTER"
MESS2="A SHIPPING WEIGHT MUST BE ENTERED"
MESS3="A UNIT PRICE MUST BE ENTERED"
MESS4="A CASE PRICE MUST BE ENTERED"
MESS5="A PRODUCT DESCRIPTION MUST BE ENTERED"
MESS6="A CATALOG # MUST BE ENTERED"
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   SUPMESS="ENTER ITEMS IN CATALOG FOR THIS SUPPLIER?"
   SYMESS="Y - USE THIS SUPPLIERS CATALOG"
   SNMESS="N - LOOK AT NEXT SUPPLIER"
   SXMESS="X - RETURN TO PREVIOUS MENU"
   MITEMNO=0
   MDESCRIPT= SPACE (60)
   MSHIPWT=0
   MUPRICE=00.00
   MCASPRICE=000.00
   MSUPNAME=SPACE(20)
   DO BACKGRD3
   DO PRODOPT
   DO CONFIRMM

   * process user's response
   DO CASE

      CASE CHR(i) $ "Aa"
        SELECT 1
        USE SUPPLIER
```

```
        DO GETSUPP
        IF XIT
          LOOP
        ENDIF
        SELECT 2
        DO ENTPROD
        LOOP

      CASE CHR(i) $ "Bb"
        SELECT 1
        USE SUPPLIER
        DO GETSUPP
        IF XIT
          LOOP
        ENDIF
        SELECT 2
        DO UPDTPRD
        LOOP

      CASE CHR(i) $ "Cc"
         EXIT
   ENDCASE
ENDDO
CLOSE ALL
RELEASE ALL
RETURN
* Eof:

* Program expenses
* Written by Gary Radke
* For Peoples Grocery Database
* Allows for entry and update of expense records
SELECT 1
USE EXPENC
CLEAR
XCORD=11
YCORD=51
ERRMESS=SPACE(60)
MESS1="THIS EXPENSE RECORD ALREADY EXISTS IN DATABASE"
MESS2="AN EXPENSE AMOUNT MUST BE ENTERED"
MESS3="AN ENTRY MUST BE MADE IN THE PAID TO SPACE"
MESS4="AN EXPENSE DESCRIPTION MUST BE ENTERED"
ERROR=.T.
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   DO EXPOPT
   MDATE=DATE()
```

```
MAMOUNT=0.00
MPAIDTO=SPACE(20)
MDESCRIPTI=SPACE(57)
MPAID=.F.
* Accept only the responses available in the menu
DO CONFIRMM

* process user's response
DO CASE
  * Add a new expense
  CASE CHR(i) $ "Aa"
    CLEAR
    DO WHILE .T.
      DO EXPFMASK
      DO ADDEXP
      DO CHKEXP
      IF ERROR
        @ 22,5 SAY ERRMESS
        WAIT
        @ 22,5
        @ 23,5
        LOOP
      ENDIF
      DO CONFIRMQ
      DO CONFIRML
      DO CASE
      * If everything is ok, save the new record
        CASE CHR(i) $ "Yy"
          APPEND BLANK
          REPLACE DATE WITH MDATE
          REPLACE AMOUNT WITH MAMOUNT
          REPLACE PAID_TO WITH MPAIDTO
          REPLACE DESCRIPTIO WITH MDESCRIPTI
          @ 22,5
          @ 23,5
          WAIT
          EXIT
      * Allow correction of errors
        CASE CHR(i) $ "Nn"
          LOOP
      * Return to the previous menu
        CASE CHR(i) $ "Xx"
          EXIT
      ENDCASE
    ENDDO

  * Review existing expense records
  CASE CHR(i) $ "Bb"
    CLEAR
    USE EXPENC
    DO WHILE .T.
      DO EXPFMASK
```

```
        DO BRSEXP
        @ 15,0 TO 21,55
        @ 16,5 SAY "Y LETS YOU UPDATE THIS EXPENSE FORM"
        @ 17,5 SAY "N SHOWS YOU THE NEXT EXPENSE FORM"
        @ 18,5 SAY "X RETURNS YOU TO THE PREVIOUS MENU"
        @ 20,5 SAY "ENTER RESPONSE"
        DO CONFIRML
        DO CASE
          CASE CHR(i) $ "Yy"
            LOOP
          CASE CHR(i) $ "Nn"
            SKIP
            IF EOF()
              @ 23,5 SAY "END OF EXPENSE FILE"
              WAIT
              GO TOP
              EXIT
            ENDIF
            LOOP
          CASE CHR(i) $  "Xx"
            EXIT
        ENDCASE
      ENDDO

    * Return to previous menu
    CASE CHR(i) $ "Cc"
        EXIT
  ENDCASE
ENDDO
USE
RETURN
* Eof:


* Program Labels
* Written by Gary Radke
* For Peoples' Grocery Database

CLEAR
XCORD=11
YCORD=51
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   DO LABOPT
   DO CONFIRMM

   * process user's response
```

```
   DO CASE

     CASE CHR(i) $ "Aa"
         DO PRINTLBL
         EXIT


     CASE CHR(i) $ "Bb"
         EXIT

     CASE CHR(i) $ "Cc"
         EXIT
   ENDCASE
ENDDO
RETURN
* Eof:

* PROGRAM BILLING
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* GENERATES AN ITEMIZED BILL FOR EACH MEMBER PARTICIPATING
* IN A GIVEN ORDER.

XCORD=11
YCORD=51
XIT=.F.
CLEAR
GO1="PREPARE BILLS FOR ORDER SUBMITTED TO "
MSUPNAME=SPACE(20)
MDATE=DATE()
MM1="WERE THERE ANY ITEMS WHICH WERE NOT DELIVERED OR "
MM2="REFUSED WHEN THE ORDER"
MM3="SUBMITTED TO "
MM4=" ON "
MM5="WAS DELIVERED.  --Y/N  TO EXIT BILLING TYPE X."
MM6="ITEM NUMBER GIVEN WAS NOT INCLUDED IN THE ORDER"
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD4
   DO BILLOPT
   DO CONFIRMM
   * process user's response
   DO CASE
     CASE CHR(i) $ "Aa"
         DO GETORD
         CLEAR
         IF XIT
           XIT=.F.
```

```
            LOOP
          ENDIF
          @ 3,5 SAY MM1
          @ 3,5+LEN(MM1) SAY MM2
          @ 4,5 SAY MM3
          @ 4,5+LEN(MM3) SAY MSUPNAME
          @ 4,5+LEN(MM3)+LEN(RTRIM(MSUPNAME)) SAY MM4
          @ 4,5+LEN(MM3)+LEN(RTRIM(MSUPNAME))+LEN(MM4) SAY MDATE
          @ 5,5 SAY MM5
          DO CONFIRML
          DO CASE
             CASE CHR(i) $ "Yy"
               DO ADJSTORD
             CASE CHR(i) $ "Xx"
               EXIT
          ENDCASE
          CLEAR
          DO PRINTBIL
      CASE CHR(i) $ "Bb"
          EXIT
   ENDCASE
ENDDO
CLOSE ALL
RELEASE ALL
RETURN
* Eof:

* PROGRAM CONSSUMM
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* GENERATES CONSIGNMENT SUMMARY REPORTS
XCORD=15
YCORD=51
YMESS="Y - PREPARE SUMMARY FOR THIS MEMBER"
NMESS="N - SKIP TO THE NEXT MEMBER"
XMESS="X - RETURN TO THE PREVIOUS MENU"
CLEAR
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD5
   DO CONSOPT
   DO WHILE .T.
       i=INKEY()
       IF UPPER(CHR(i))$"ABCD"
            EXIT
       ENDIF
       i=0
       @ XCORD,YCORD SAY UPPER(CHR(i))
```

```
   ENDDO

   * process user's response
   DO CASE

     CASE CHR(i) $ "Aa"
         DO SOLDUPD

     CASE CHR(i) $ "Bb"
         DO ALLUSC

     CASE CHR(i) $ "Cc"
         DO ALLC

     CASE CHR(i) $ "Dd"
         EXIT
   ENDCASE
ENDDO
CLOSE ALL
RELEASE ALL
RETURN
* Eof:

* Program Storeord
* Written by Gary Radke
* For Peoples Grocery Database
* Prints a summary of a given combined store order

XIT=.F.
CLEAR
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD3
   DO STOROPT
   DO WHILE .T.
      i=INKEY()
      IF UPPER(CHR(i))$"AB"
         EXIT
      ENDIF
      i=0
      @ 15,58 SAY UPPER(CHR(i))
   ENDDO

   * process user's response
   DO CASE

     CASE CHR(i) $ "Aa"
         DO COMBORD
```

```
      CASE CHR(i) $ "Bb"
          EXIT

   ENDCASE
ENDDO
RETURN
* Eof:


* Program Finances
* Written by Gary Radke
* For Peoples Grocery Database

CLEAR
DO WHILE .T.
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command

   * Clear the screen and display the main menu
   CLEAR
   DO BACKGRD5
   DO FINOPT
   XCORD=15
   YCORD=51
   DO WHILE .T.
      i=INKEY()
      IF UPPER(CHR(i))$"ABCD"
         EXIT
      ENDIF
      i=0
      @ XCORD,YCORD SAY UPPER(CHR(i))
   ENDDO

   * process user's response
   DO CASE

     CASE CHR(i) $ "Aa"
         DO LISTEX

     CASE CHR(i) $ "Bb"
         DO EXP_SAL

     CASE CHR(i) $ "Cc"
         DO CONEXP

     CASE CHR(i) $ "Dd"
         EXIT
   ENDCASE
ENDDO
CLOSE ALL
RELEASE ALL
```

```
RETURN
* Eof:

*PROGRAM BACKGRD3
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*PRODUCES A GENERIC BACKGROUND FOR A MENU WITH THREE OPTIONS.
CLEAR
@1,1 TO 12,77
@3,3 TO 10,75 DOUBLE
@ 2,2 SAY REPLICATE (CHR (176), 75)
@3,2 SAY CHR (176)
@ 4,2 SAY CHR (176)
@ 5,2 SAY CHR (176)
@ 6,2 SAY CHR (176)
@ 7,2 SAY CHR (176)
@ 8,2 SAY CHR (176)
@ 9,2 SAY CHR (176)
@ 10,2 SAY CHR (176)
@ 3,76 SAY CHR (176)
@ 4,76 SAY CHR (176)
@ 5,76 SAY CHR (176)
@ 6,76 SAY CHR (176)
@ 7,76 SAY CHR (176)
@ 8,76 SAY CHR (176)
@ 9,76 SAY CHR (176)
@ 10,76 SAY CHR (176)
@ 11,2 SAY REPLICATE (CHR (176), 75)
RETURN
*Eof;

*PROGRAM MOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR ADDING A NEW MEMBER TO OR UPDATING
*INFORMATION ON A CURRENT MEMBER
@ 4,27 SAY " A. ADD A NEW MEMBER"
@ 6,27 SAY " B. UPDATE A CURRENT MEMBER"
@ 8,27 SAY " C. RETURN TO MAIN MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof;

* PROGRAM MEMDUP
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* CHECKS FOR DUPLICATE MEMBER RECORDS BY SEARCHING FOR DUPLICATE
* MEMBER # S WHEN A NEW MEMBER IS ADDED
SET TALK OFF
SELECT 1
USE MEMBER INDEX MEMNO
SEEK MMEMNO
```

```
IF FOUND() THEN
  CLEAR
  @ 3,5 SAY DUPNUM
  @ 3,5 + LEN(DUPNUM) SAY MMEMNO
  @ 4,5 SAY DUPMESS
  WAIT
ENDIF
RETURN
*Eof

* PROGRAM CONFIRMM.PRG
* USED IN PEOPLES GROCERY DATABASE
* CONFIRMS RESPONSE TO VARIOUS MENU CHOICES IN THE
* PROGRAM.  ONLY ALLOWS A, B, OR C AS A RESPONSE

DO WHILE .T.
   i=INKEY()
   IF UPPER(CHR(i))$"ABC"
      EXIT
   ENDIF
   i=0
   @ XCORD,YCORD SAY UPPER(CHR(i))
ENDDO
RETURN
* Eof:

* PROGRAM NEWMEM
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ENTERS A NEW MEMBER RECORD INTO THE MEMBER DBF
CLEAR
DO WHILE .T.
   * Display membership form
   DO MEMFMASK
   DO ADDMEM
   DO CHKMEM
   IF ERR
     @ 22,5 SAY ERRMESS
     WAIT
     @ 22,0
     @ 23,0
     LOOP
   ENDIF
   * Accept only allowed responses
   DO CONFIRMQ
   DO CONFIRML
   DO CASE
      * Add the record to the db
      CASE CHR(i) $ "Yy"
        APPEND BLANK
        REPLACE MEM_NO WITH MMEMNO
        REPLACE FIRSTNAME WITH MFIRST
```

```
        REPLACE LASTNAME WITH MLAST
        REPLACE STREET WITH MSTREET
        REPLACE TOWN WITH MCITY
        REPLACE ZIPCODE WITH MZIP
        REPLACE STATE WITH MSTATE
        REPLACE HOMEPHONE WITH MHOME
        REPLACE WORKPHONE WITH MWORK
        INDEX ON MEM_NO TO MEMNO
        @ 22,0
        @ 23,0
        WAIT
        EXIT

      * Allow corrections
      CASE CHR(i) $ "Nn"
        LOOP
      * Abandon operation
      CASE CHR (i) $ "Xx"
        EXIT
    ENDCASE
  ENDDO
RETURN
*Eof

* PROGRAM MEMFMASK.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR MEMBER INFORMATION FORM
CLEAR
@ 0,0 TO 21,79 DOUBLE
@ 2,23 TO 5,53
@ 3,25 SAY "PEOPLES GROCERY COOPERATIVE"
@ 4, 25 SAY "MEMBERSHIP INFORMATION FORM"
@ 6,9 SAY "MEMBER #"
@ 8,9 SAY "FIRST NAME"
@ 8,30 SAY "LAST NAME"
@ 10,5 SAY "ADDRESS"
@ 12,9 SAY "STREET"
@ 13,9 SAY "CITY"
@ 14,9 SAY "STATE"
@ 14,18 SAY "ZIP CODE"
@ 17,9 SAY "HOME PHONE"
@ 18,9 SAY "WORK PHONE"
RETURN
*Eof

*PROGRAM ADDMEM.PRG
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*READS MEMBER INFORMATION INPUT

@6,18 GET MMEMNO
```

```
@8,20 GET MFIRST
@ 8,41 GET MLAST
@ 12,16 GET MSTREET
@ 13,14 GET MCITY
@ 14,15 GET MSTATE
@ 14,27 GET MZIP
@ 17,20 GET MHOME
@ 18,20 GET MWORK
READ
RETURN
* Eof:

* PROGRAM CHKMEM
* WRITTEN BY GARY RADKE
* CHECKS FOR ERRORS WHEN A NEW MEMBER RECORD IS ENTERED
ERR=.T.
DO CASE
   * Check to see if this member # is to large
   CASE (MMEMNO > 999)
     ERRMESS=MESS1

   * Make sure a member # has been entered
   CASE (MMEMNO < 1) THEN
     ERRMESS=MESS2

   *Make sure a member name has been entered
   CASE (LEN(RTRIM(MFIRST)) < 1)
     ERRMESS=MESS3

   OTHERWISE
     ERR=.F.
ENDCASE
* Check to see if this member # already exists in db
SEEK MMEMNO
* If it exists, use another
IF FOUND() THEN
  ERRMESS=DUPMESS
  ERR=.T.
ENDIF
RETURN
*Eof

*PROGRAM CONFIRMQ.PRG
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
@ 22,5 SAY "IS THIS INFORMATION CORRECT? Y/N"
@ 23,5 SAY "TYPE 'X' TO ABANDON OPERATION"
RETURN
* Eof:

*PROGRAM CONFIRML.PRG
*WRITTEN BY GARY RADKE
```

```
*FOR PEOPLES GROCERY DATABASE
*ACCEPTS ONLY Y, N, OR X AS A RESPONSE TO PROGRAM
*QUERIES
DO WHILE .T.
   i=INKEY()
   IF UPPER(CHR(i))$"YNX"
      EXIT
   ENDIF
   i=0
   @ 22,53 SAY UPPER(CHR(i))
ENDDO
@ 22,5
@ 23,5
RETURN
*Eof

*PROGRAM MEMUOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR ADDING A NEW MEMBER TO OR UPDATING
*INFORMATION ON A CURRENT MEMBER
@ 4,27 SAY " A. ENTER MEMBER #"
@ 6,27 SAY " B. BROWSE MEMBER FILE"
@ 8,27 SAY " C. RETURN TO PREVIOUS MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof

*PROGRAM GETMEM.PRG
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*LOCATES A MEMBERS MEMBERSHIP RECORD SO INFORMATION FROM IT
*MAY BE USED ELSEWHERE IN THE APPLICATION
XIT=.F.
MEMNAME=SPACE(20)
MMEMNO=0
CLEAR
* Step throuhg the member info db and display
* member # and name for each record examined
IF (.NOT. EOF())
  DO WHILE .T.
    MEMNAME= RTRIM(FIRSTNAME) + " " + RTRIM(LASTNAME)
    @ 3,5 SAY "MEMBER #"
    @ 3,14 SAY MEM_NO
    @ 3,25 SAY "NAME"
    @ 3,30 SAY MEMNAME
    @ 5,5 SAY MEMESS
    @ 7,0 TO 13,55 DOUBLE
    @ 8,5 SAY YMESS
    @ 10,5 SAY NMESS
    @ 12,5 SAY XMESS
    DO CONFIRML
```

```
    DO CASE
       * Use this record as a reference
       * or in an operation
       CASE CHR(i) $ "Yy"
         MMEMNO=MEM_NO
         EXIT
       * Ignore this record look at the next
       CASE CHR(i) $ "Nn"
         SKIP
         IF EOF()
           @ 23,5 SAY "END OF MEMBER FILE"
           WAIT
           XIT=.T.
           EXIT
         ELSE
           @ 3,30
           LOOP
         ENDIF
       * Abandon operation
       CASE CHR(i) $  "Xx"
         XIT=.T.
         EXIT
    ENDCASE
  ENDDO
ENDIF
RETURN
* Eof:

*PROGRAM UPMEMRD.PRG
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*READS MEMBERSHIP UPDATE INFORMATION

@ 6,18 SAY MEM_NO
@ 8,20 SAY FIRSTNAME
@ 8,41 SAY LASTNAME
@ 12,16 GET MSTREET
@ 13,14 GET MCITY
@ 14,15 GET MSTATE
@ 14,27 GET MZIP
@ 17,20 GET MHOME
@ 18,20 GET MWORK
READ
RETURN
* Eof:

* PROGRAM UPDTMEM
* WRITTEN BY GARY RADKEA
* FOR PEOPLESM GROCERY DATABASE
* ALLOWS UPDATE OF EXISTING MEMBER RECORDS

SELECT 1
```

```
DO WHILE .T.
   * Clear screen and display menu
   CLEAR
   DO BACKGRD3
   DO MEMUOPT
   * Accept only allowed responses
   DO CONFIRMM

   DO CASE
      * Enter a members number and locate that record
      CASE CHR(i) $ "Aa"
        DO WHILE .T.
           CLEAR
           MMEMNO=0
           @ 3,5 SAY "ENTER MEMBERS NUMBER"
           @ 3,26 GET MMEMNO
           READ
           SEEK MMEMNO
           IF .NOT. FOUND()
              @ 4,5 SAY "MEMBER # DOES NOT EXIST IN DATABASE"
              @ 5,5 SAY "TRY ANOTHER"
              WAIT
              LOOP
           ELSE
              EXIT
           ENDIF
        ENDDO
        CLEAR

      * Browse through the db to locate a record
      CASE CHR(i) $ "Bb"
        CLEAR
        GO TOP
        DO GETMEM
        IF XIT
          LOOP
        ENDIF
      * Abandon operation
      CASE CHR(i) $"Cc"
        EXIT
   ENDCASE
   IF CHR(i) $ "Xx"
     LOOP
   ENDIF
   * Initialize temp variables with chosen record
   * values
   MSTREET = STREET
   MCITY = TOWN
   MSTATE = STATE
   MZIP = ZIPCODE
   MHOME = HOMEPHONE
   MWORK = WORKPHONE
```

```
    DO WHILE .T.
      * Display membership form and get new info
      DO MEMFMASK
      DO UPMEMRD
      DO CONFIRMQ
      * Accept only allowed responses
      DO CONFIRML
      DO CASE
        * Replace record values with new values
        CASE CHR(i) $ "Yy"
          REPLACE STREET WITH MSTREET
          REPLACE TOWN WITH MCITY
          REPLACE ZIPCODE WITH MZIP
          REPLACE STATE WITH MSTATE
          REPLACE HOMEPHONE WITH MHOME
          REPLACE WORKPHONE WITH MWORK
          @ 22,0
          @ 23,0
          WAIT
          EXIT
        * Allow corrections
        CASE CHR(i) $ "Nn"
          LOOP
        * Abandon operations
        CASE CHR (i) $ "Xx"
          EXIT
      ENDCASE
    ENDDO
ENDDO
RETURN
*Eof;

*PROGRAM ORDOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR UPDATING THE MEMBER ORDER DATABASE
@ 4,27 SAY " A. ENTER A MEMBERS ORDER"
@ 6,27 SAY " B. UPDATE AN EXISTING ORDER"
@ 8,27 SAY " C. RETURN TO MAIN MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof;

*PROGRAM NEWORD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ENTERS ITEM ORDERS FOR COOP MEMBERS
* Select a supplier to order from
SELECT 4
GO TOP
DO GETSUPP
IF XIT
```

```
  RETURN
ENDIF
MSUPNAME=SUPNAME
DO WHILE .T.
   * Display menu to choose member to order for
   MMEMNO=0
   MITEMNO=0
   MQUANTITY=0
   DO BACKGRD3
   DO MORDOPT
   DO CONFIRMM
   DO CASE
      * Enter a known member #
      CASE CHR (i) $ "Aa"
        DO WHILE .T.
          CLEAR
          @ 3,5 SAY QUERY1
          @ 3,5+LEN(QUERY1) GET MMEMNO
          READ
          * Make sure given member # exists in db
          SELECT 1
          SEEK MMEMNO
          IF (.NOT. FOUND()) THEN
            @ 5,5 SAY "MEMBER # DOES NOT EXIST IN"
            @ 6,5 SAY "DATABASE, TRY ANOTHER"
            WAIT
            LOOP
          ENDIF
          MEMNAME = RTRIM (FIRSTNAME) + " " + RTRIM (LASTNAME)
          CLEAR
          EXIT
        ENDDO
      * Browse member file
      CASE CHR(i) $ "Bb"
        SELECT 1
        GO TOP
        DO GETMEM
        IF XIT
           LOOP
        ENDIF
        MMEMNO=MEM_NO
      * Return to first menu in this application
      CASE CHR(i) $ "Cc"
        EXIT
   ENDCASE
   * Display an order form as long as the user wants to
   * submit orders
   DO WHILE .T.
      DO MOFMASK
      @ 7,14 SAY MMEMNO
      @ 7,65 SAY MDATE
      @ 9,17 SAY MEMNAME
```

```
@ 9,49 SAY MSUPNAME
@ 13,6 GET MITEMNO
@ 20,26 SAY LASTITEM
READ
SELECT 2
LOCATE FOR MSUPNAME=SUPNAME .AND. MITEMNO=ITEMNO
* Tell user if requested item is not in the product
* catalog for this supplier
IF (.NOT. FOUND()) THEN
   @ 22,5 SAY MESS1
   WAIT
   @ 22,0
   @ 23,0
     LOOP
ENDIF
MDESCRIPT = RTRIM(DESCRIPT)
@ 17,9 SAY MDESCRIPT
@ 13,49 GET MQUANTITY
READ
DO CHKORD
IF ERR
   @ 22,5 SAY ERRMESS
   WAIT
   @ 22,0
   @ 23,0
   LOOP
ENDIF
DO CONFIRMQ
DO CONFIRML
DO CASE
   CASE CHR(i) $ "Yy"
     * Make sure order is not a duplicate
     * Create a new order record and enter given values
     APPEND BLANK
     REPLACE SUPNAME WITH MSUPNAME
     REPLACE MEM_NO WITH MMEMNO
     REPLACE ITEMNO WITH MITEMNO
     REPLACE QUANTITY WITH MQUANTITY
     REPLACE DATE WITH MDATE
     @ 22,0
     @ 23,0
     INDEX ON DTOC(DATE) + SUPNAME TO STORORD
     @ 22,5 SAY "ORDER ANOTHER ITEM FOR THIS MEMBER? Y/N "
     DO CONFIRML
     @ 22,0
     IF UPPER(CHR(i)) $ "Y"
        LASTITEM=MDESCRIPT
        MITEMNO=0
        MQUANTITY=0
        LOOP
     ENDIF
     EXIT
```

```
          * If a mistake was made, let user correct it
        CASE CHR(i) $ "Nn"
          LOOP
          * Return to previous menu
        CASE CHR(i) $ "Xx"
          CLEAR
          EXIT
      ENDCASE
    ENDDO
  ENDDO
RETURN
*Eof

*PROGRAM GETSUPP
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*BROWSE THROUGH SUPPLIERS
* DO WHILE .T. means DO WHILE TRUE  i.e. DO FOREVER
* The DO WHILE will be terminated by an EXIT command
XIT=.F.
CLEAR
IF (.NOT. EOF())
  DO WHILE .T.
    MSUPNAME= RTRIM (SUPNAME)
    @ 3,5 SAY "SUPPLIER"
    @ 3,15 SAY MSUPNAME
    @ 5,5 SAY SUPMESS
    @ 7,0 TO 13,55 DOUBLE
    @ 8,5 SAY SYMESS
    @ 10,5 SAY SNMESS
    @ 12,5 SAY SXMESS
    DO CONFIRML
    DO CASE
       CASE CHR(i) $ "Yy"
         MSUPNAME=SUPNAME
         EXIT
       CASE CHR(i) $ "Nn"
         SKIP
         IF EOF()
           @ 23,5 SAY "END OF SUPPLIER FILE"
           WAIT
           XIT=.T.
           EXIT
         ELSE
           @ 3,15
           LOOP
         ENDIF
       CASE CHR(i) $  "Xx"
         XIT=.T.
         EXIT
    ENDCASE
  ENDDO
```

```
ENDIF
RETURN
* Eof:

*PROGRAM MORDOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR ENTERING A MEMBER ORDER
@ 4,27 SAY " A. ENTER KNOWN MEMBER #"
@ 6,27 SAY " B. BROWSE MEMBER LIST TO FIND MEMBER # "
@ 8,27 SAY " C. RETURN TO PREVIOUS MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof

* PROGRAM MOFMASK.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR MEMBER ORDER FORM
CLEAR
@ 0,0 TO 21,79 DOUBLE
@ 2,31 TO 4,49
@ 3,32 SAY "MEMBER ORDER FORM"
@ 7,5 SAY "MEMBER #"
@ 7,60 SAY "DATE"
@ 9,5 SAY "MEMBER NAME"
@ 9,40 SAY "SUPPLIER"
@ 11,7 SAY "CATALOG #"
@ 15,32 SAY "ITEM DESCRIPTION"
@ 16,8 TO 18,71
@ 11,50 SAY "NUMBER OF UNITS"
@ 12,38 SAY "(# OF POUNDS, OUNCES, BAGS, CANS ETC.)"
@ 19,5 SAY "LAST ITEM ORDERED"
@ 20,5 SAY "FOR MEMBER THIS DATE"
RETURN
* Eof

* PROGRAM CHKORD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* CHECKS FOR ORDER ENTRY ERRORS
ERR=.T.
SELECT 3
* Check to see if there is a duplicate order
DO ORDDUP
DO CASE
   * Require the order of at least one item unit
   CASE (MQUANTITY < 1)
     ERRMESS= QUANTM
   CASE FOUND()
     ERRMESS=DUPMESS
   OTHERWISE
```

```
     ERR=.F.
ENDCASE
RETURN
*Eof;

* PROGRAM ORDDUP
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* CHECKS AN ENTERED ORDER TO SEE IF IT IS A DPULICATE
* OF AN ORDER ENTERED EARLIER

DUP = .T.
LOCATE FOR MMEMNO=MEM_NO .AND. MITEMNO=ITEMNO .AND. MDATE=DATE .AND.
QUANTITY=MQUANTITY
IF  EOF() THEN
  DUP =.F.
ENDIF
RETURN
*Eof


*PROGRAM UPDTORD
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*PROGRAM UPDATES AN EXISTING MEMBER ORDER
DO WHILE .T.
   * Select a supplier
   SELECT 4
   GO TOP
   DO GETSUPP
   IF XIT
     EXIT
   ENDIF
   MSUPNAME=SUPNAME
   DO WHILE .T.
      * Display menu for selecting a member
      DO BACKGRD3
      DO MORDOPT
      DO CONFIRMM
      DO CASE
         CASE CHR (i) $ "Aa"
           CLEAR
           * Enter known member
           DO WHILE .T.
              @ 3,5 SAY QUERY4
              @ 3,5+LEN(QUERY4) GET MMEMNO
              READ
              SELECT 1
              * See if member # exists in db
              SEEK MMEMNO
              IF (.NOT. FOUND()) THEN
                @ 5,5 SAY "MEMBER # DOES NOT EXIST IN"
```

```
            @ 6,5 SAY "DATABASE, TRY ANOTHER"
            WAIT
            @ 5,0
            @ 6,0
            LOOP
          ENDIF
          MEMNAME = RTRIM (FIRSTNAME) + " " + RTRIM(LASTNAME)
          CLEAR
          EXIT
        ENDDO

      * Browse member file
      CASE CHR(i) $ "Bb"
        SELECT 1
        GO TOP
        DO GETMEM
        MMEMNO=MEM_NO
        IF XIT THEN
           EXIT
        ENDIF
      * Return to first menu in this application
      CASE CHR(i) $ "Cc"
        EXIT
ENDCASE
DO WHILE .T.
  * Display order form & let user enter date and itemno
  DO MOFMASK
  @ 7,14 SAY MMEMNO
  @ 7,65 GET MDATE
  @ 9,17 SAY MEMNAME
  @ 9,49 SAY MSUPNAME
  @ 13,6 GET MITEMNO
  READ
  * See if given order exists in order file
  SELECT 3
  LOCATE FOR MSUPNAME=SUPNAME .AND. MITEMNO=ITEMNO .AND.
  MDATE=DATE
  IF EOF()
    @ 22,5 SAY "THIS MEMBER HAS NOT ORDERED THIS ITEM ON GIVEN
    DATE"
    WAIT
    @ 22,0
    @ 23,0
    @ 22,5 SAY "ABANDON UPDATE THIS MEMBER? Y/N"
    DO CONFIRML
    IF (UPPER(CHR(i)) = "Y")
      EXIT
    ENDIF
    @ 22,0
    @ 23,0
    LOOP
  ENDIF
```

```
   * Find ordered product  so the description can be
   * displayed
   SELECT 2
   LOCATE FOR MSUPNAME=SUPNAME .AND. MITEMNO=ITEMNO
   MDESCRIPT = RTRIM(DESCRIPT)
   SELECT 3
   @ 17,9 SAY MDESCRIPT
   @ 13,49 GET MQUANTITY
   READ
   * Require the user to order some of the selected
   * item
   IF (MQUANTITY < 1) THEN
     @ 22,5 SAY QUANTM
     WAIT
     @ 22,5
     @ 23,5
     LOOP
   ENDIF
   DO CONFIRMQ
   DO CONFIRML
   @ 22,0
   @ 23,0
   DO CASE
      * If entered info is correct then store it
      CASE CHR(i) $ "Yy"
        SELECT 3
        REPLACE SUPNAME WITH MSUPNAME
        REPLACE MEM_NO WITH MMEMNO
        REPLACE ITEMNO WITH MITEMNO
        REPLACE QUANTITY WITH MQUANTITY
        REPLACE DATE WITH MDATE
        @ 22,5 SAY "UPDATE ANOTHER ITEM FOR THIS MEMBER? Y/N "
        DO CONFIRML
        @ 22,0
        IF CHR(i) $ "Yy"
          LASTITEM=MDESCRIPT
          MITEMNO=0
          MQUANTITY=0
          LOOP
        ENDIF
        EXIT
      CASE CHR(i) $ "Nn"
        EXIT
        *LOOP
      CASE CHR(i) $ "Xx"
        EXIT
   ENDCASE
ENDDO
* Repeat the process with a new member?
@ 22,0
@ 22,5 SAY "UPDATE ORDER FOR A NEW MEMBER? Y/N"
DO CONFIRML
```

```
      DO CASE
         CASE CHR(i) $ "Yy"
           LOOP
         CASE CHR(i) $ "Nn"
           EXIT
         CASE CHR(i) $ "Xx"
      ENDCASE
   ENDDO
   IF (UPPER(CHR(i))="C")
      EXIT
   ENDIF
   @ 22,5
   @ 22,5 SAY "UPDATE AN ORDER WITH A NEW SUPPLIER? Y/N"
   DO CONFIRML
   DO CASE
      CASE CHR(i) $ "Yy"
        LOOP
      CASE CHR(i) $ "Nn"
        EXIT
      CASE CHR(i) $ "Xx"
        EXIT
   ENDCASE
ENDDO
RETURN
*Eof

*PROGRAM CONOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR MAINTAINING THE CONSIGNMENTS DATABASE
@ 4,27 SAY " A. ADD NEW CONSIGNMENTS"
@ 6,27 SAY " B. UPDATE EXISTING CONSIGNMENTS"
@ 8,27 SAY " C. RETURN TO MAIN MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof

* PROGRAM ECON
* WRITTEN BY GARY RADKE
* FOR PEOPLESM GROCERY DATABASE'
* ENTERS A NEW CONSIGNMENT INTO THE CONSIGNMENT DBF

* continue adding consignments until user is done
DO WHILE .T.
  CLEAR
  DO CONFMASK
  DO ADDCON
  DO CHECKCON
  IF ERROR
    @ 21,5 SAY ERRORMESS
    WAIT
    @ 21,0
```

```
    @ 22,0
    LOOP
  ENDIF
  DO CONFIRMQ
  DO CONFIRML
  SELECT 2
  DO CASE
     * append this record to consignments
     CASE CHR(i) $ "Yy"
       APPEND BLANK
       REPLACE MEM_NO WITH MMEMNO
       REPLACE DESCRIPT WITH MDESCRIPT
       REPLACE NO_OF_ITEM WITH MNUMITEM
       REPLACE CONSGRPRIC WITH MPRICE
       REPLACE ALL_SOLD WITH MSOLD
       REPLACE DATE WITH MDATE
       REPLACE MEMPAID WITH MPAID
       @ 21,0
       @ 22,0
       INDEX ON MEM_NO TO CMEMNO
       * let the user change incorrect information
     CASE CHR(i) $ "Nn"
       @ 21,0
       @ 22,0
       LOOP
       * abandon consignment entry without adding anything
     CASE CHR(i) $"Xx"
       EXIT
  ENDCASE
  * let user add as many consignments as they want
  @ 22,5 SAY "ENTER ANOTHER CONSIGNMENT Y/N"
  DO CONFIRML
  IF CHR(i) $ "Yy"
    * reset the memory variables in preperation of next
    * consignment entry otherwise exit
    CLEAR
    MMEMNO=0
    MEMNAME=SPACE(40)
    MDESCRIPT=SPACE(20)
    MNUMITEM=0
    MPRICE=000.00
    MSOLD=.F.
    MPAID=.F.
    MDATE=DATE()
    LOOP
  ENDIF
  EXIT
ENDDO
RETURN
*Eof
```

```
* PROGRAM CCON
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ALLOWS UPDATE OF AN EXISTING CONSIGNMENT RECORD
XIT=.F.
DO WHILE .T.
  DO BACKGRD3
  DO UPCOPT
  DO CONFIRMM
  SELECT 1
  GO TOP
  DO CASE
     * Enter a known member # and update their consignment
     CASE CHR(i) $ "Aa"
       DO WHILE .T.
         CLEAR
         @ 3,5 SAY "UPDATE CONSIGNMENT FOR MEMBER #"
         @ 3,37 GET MMEMNO
         READ
         SELECT 1
         SEEK MMEMNO
         IF((.NOT. FOUND()).OR. (MMEMNO < 1))
           @ 22,5 SAY MESS4
           WAIT
           @ 22,0
           @ 23,0
           LOOP
         ENDIF
         MEMNAME= RTRIM (FIRSTNAME) + " " + RTRIM(LASTNAME)
         SELECT 2
         SEEK MMEMNO
         IF (.NOT. FOUND())
           @ 22,5 SAY MESS5
           WAIT
           @ 22,0
           @ 23,0
           LOOP
         ENDIF
         EXIT
       ENDDO
       DO CONUPD
       * Browse through the member file to find the right one
     CASE CHR(i) $ "Bb"
       DO WHILE .T.
         CLEAR
         SELECT 1
         DO GETMEM
         DO CASE
            CASE CHR(i) $ "Yy"
              SELECT 2
              SEEK MMEMNO
```

```
            IF (.NOT. FOUND())
              @ 22,5 SAY MESS5
              WAIT
              @ 22,0
              @ 23,0
              SELECT 1
              LOOP
            ENDIF
            DO CONUPD
            IF XIT
              LOOP
            ENDIF
            EXIT
          CASE CHR(i) $  "Xx"
            EXIT
        ENDCASE
      ENDDO
      * Exit and return to main menu this module
      CASE CHR(i) $ "Cc"
        EXIT
    ENDCASE
ENDDO
RETURN
*Eof

* PROGRAM CONFMASK.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR PRODUCT CONSIGNMENT FORM
CLEAR
@ 0,0 TO 20,79 DOUBLE
@ 2,25 TO 4,53
@ 3,27 SAY "PRODUCT CONSIGNMENT FORM"
@ 7,9 SAY "MEMBER #"
@ 7,60 SAY "DATE"
@ 10,5 SAY "PRODUCT DESCRIPTION"
@ 10,27 SAY "NUMBER PLACED"
@ 11,27 SAY "ON CONSIGNMENT"
@ 10,45 SAY "PRICE / ITEM PAID"
@ 11,49 SAY "TO MEMBER"
@ 17,10 SAY "ALL ITEMS SOLD"
@ 17,55 SAY "MEMBER PAID"
RETURN
*Eof

*PROGRAM ADDCON
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*PROGRAM READS CONSIGNMENT ENTRY INFORMATION
@ 7,19 GET MMEMNO
@ 7,65 GET MDATE
@ 13,5 GET MDESCRIPT
```

```
@ 13,32 GET MNUMITEM
@ 13,50 GET MPRICE
@ 17,25 GET MSOLD
@ 17,68 GET MPAID
READ
RETURN
* Eof:


* PROGRAM CHECKCON
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* MAKES SURE VALUES HAVE BEEN ENTERED IN FIELDS
* THAT REQUIRE A VALUE TO MAKE THE RECORD VALID
* ASSIGNS AN ERROR MESSAGE IF AN INCORRECT VALUE IS FOUND.
ERROR=.T.
SELECT 1
SEEK MMEMNO
DO CASE
  CASE (LEN(RTRIM(MDESCRIPT)) < 1)
    ERRORMESS=MESS1
  CASE (MNUMITEM < 1)
    ERRORMESS=MESS2
  CASE (MPRICE < 0.01)
    ERRORMESS=MESS3
  CASE (.NOT. FOUND() .OR. MMEMNO < 1)
    ERRORMESS=MESS4
  OTHERWISE
    ERROR=.F.
ENDCASE
SELECT 2
LOCATE FOR MMEMNO = MEM_NO .AND. RTRIM(DESCRIPT) = RTRIM(MDESCRIPT)
.AND. MNUMITEM = NO_OF_ITEM .AND. MPRICE = CONSGRPRICE .AND.
DATE=MDATE
IF FOUND()
  ERRORMESS=MESS6
  ERROR=.T.
ENDIF
SELECT 2
RETURN
* Eof

*PROGRAM UPCOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR MAINTAINING THE CONSIGNMENTS DATABASE
@ 4,27 SAY " A. UPDATE CONSIGNMENTS WHERE MEMBER # IS KNOWN"
@ 6,27 SAY " B. FIND MEMBER # TO UPDATE CONSIGNMENTS"
@ 8,27 SAY " C. RETURN TO PREVIOUS MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof
```

```
*PROGRAM CONUPD
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*UPDATES AN EXISTING CONSIGNMENT RECORD

* Use the consignment dbf
* Locate the first record for the selected member #
LOCATE FOR MMEMNO=MEM_NO
DO WHILE .T.
   CLEAR
   * See if this is the record the user wants to update
   DO CONBMSK
   DO CONFIRML
   DO CASE
      CASE CHR(i) $ "Yy"
        CLEAR
        * Allow update of the chosen record
        DO CONFMASK
        CONNUM=RECNO()
        MNUMITEM=NO_OF_ITEM
        MPRICE=CONSGRPRIC
        MSOLD=ALL_SOLD
        MPAID=MEMPAID
        MDESCRIPT=DESCRIPT
        DO WHILE .T.
          SELECT 2
          DO UPCONRD
          * Check for missing values
          DO CHECKCON
          * Ignore duplicate error otherwise print error message
          DO CASE
             CASE(ERRORMESS=MESS2)
               @ 21,5 SAY ERRORMESS
               WAIT
               @ 21,0
               @ 22,0
               GO CONNUM
               LOOP
             CASE(ERRORMESS=MESS3)
               @ 21,5 SAY ERRORMESS
               WAIT
               @ 21,0
               @ 22,0
               GO CONNUM
               LOOP
             OTHERWISE
          ENDCASE

          DO CONFIRMQ
          DO CONFIRML
```

```
          DO CASE
            * Save changes
            CASE CHR(i) $ "Yy"
              REPLACE NO_OF_ITEM WITH MNUMITEM
              REPLACE CONSGRPRIC WITH MPRICE
              REPLACE ALL_SOLD WITH MSOLD
              REPLACE DATE WITH MDATE
              REPLACE MEMPAID WITH MPAID
              @ 21,5
              @ 22,5
              @ 23,0
              XIT=.T.
              EXIT
            * Allow corrections
            CASE CHR(i) $ "Nn"
              @ 21,0
              @ 22,0
              @ 23,0
              GO CONNUM  && THE CONSIGNMENT RECORD BEING OPERATED ON
              LOOP

            CASE CHR(i) $ "Xx"
              @ 21,0
              @ 22,0
              EXIT
          ENDCASE
          XIT=.T.
          EXIT
        ENDDO

        * If this isn't the one to update, look at the next
        CASE CHR(i) $ "Nn"
          CONTINUE
          IF EOF()
            @ 22,5 SAY " END OF CONSIGNMENT RECORDS FOR THIS MEMBER"
            WAIT
            EXIT
          ELSE
            @ 8,29
            @ 8,29 SAY DESCRIPT
            LOOP
          ENDIF
        * Exit update loop
        CASE CHR(i) $ "Xx"
        XIT=.T.
        EXIT
      ENDCASE
    * Return to consignment program
    IF ((UPPER(CHR(i)) $ "X") .OR. (XIT))
       EXIT
    ENDIF
  ENDDO
```

```
 GO TOP
 RETURN
* Eof:


*PROGRAM UPCONRD
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*TEMPLATE FOR INPUT TO AN EXISTING CONSIGNMENT RECORD

@ 7,19 SAY MEM_NO
@ 7,65 SAY DATE
@ 8,11 SAY MEMNAME
@ 13,5 SAY DESCRIPT
@ 13,32 GET MNUMITEM
@ 13,50 GET MPRICE
@ 17,25 GET MSOLD
@ 17,68 GET MPAID
READ
RETURN
* Eof:


*PROGRAM CONBMSK
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
*MASK FOR BROWSING THROUGH A MEMBERS CONSIGNMENT
*RECORDS TO FIND THE ONE TO UPDATE
@ 3,5 SAY "MEMBER #"
@ 3,50 SAY "DATE"
@ 6,5 SAY "MEMBER NAME"
@ 8,5 SAY "CONSIGNMENT DESCRIPTION"
@ 3,14 SAY MEM_NO
@ 3,56 SAY DATE
@ 6,17 SAY MEMNAME
@ 8,29 SAY DESCRIPT
@ 10,5 SAY "UPDATE THIS CONSIGNMENT RECORD? Y/N"
@ 13,0 TO 21,60 DOUBLE
@ 14,5 SAY "TYPING Y RETREIVES THIS RECORD"
@ 16,5 SAY "TYPING N LOCATES THE NEXT CONSIGNMENT RECORD FOR"
@ 17,5 SAY "THIS MEMBER"
@ 19,5 SAY "TYPING X RETURNS YOU TO THE PREVIOUS MENU"
RETURN
* Eof:

*PROGRAM SUPPOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR UPDATING THE SUPPLIERS DATABASE
@ 4,27 SAY " A. ADD A NEW SUPPLIER"
@ 6,27 SAY " B. UPDATE A CURRENT SUPPLIERS RECORD"
@ 8,27 SAY " C. RETURN TO MAIN MENU"
```

```
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof

* PROGRAM SUPFMASK.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR SUPPLIER INFORMATION FORM
CLEAR
@ 0,0 TO 21,79 DOUBLE
@ 2,23 TO 5,53
@ 3,25 SAY "PEOPLES GROCERY COOPERATIVE"
@ 4, 26 SAY "SUPPLIER INFORMATION FORM"
@ 7,9 SAY "SUPPLIER NAME"
@ 9,9 SAY "NAME OF CONTACT"
@ 12,5 SAY "ADDRESS"
@ 14,9 SAY "STREET"
@ 15,9 SAY "CITY"
@ 16,9 SAY "STATE"
@ 16,18 SAY "ZIP CODE"
@ 19,9 SAY "PHONE"
RETURN
*Eof

*PROGRAM ADDSUP
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*ENTERS SUPPLIER INFORMATION INTO SUPPLIER.DBF
MSUPNAME=SPACE(20)
@ 7,23 GET MSUPNAME
@ 9,25 GET MCONTACT
@ 14,16 GET MSTREET
@ 15,14 GET MCITY
@ 16,15 GET MSTATE
@ 16,27 GET MZIP
@ 19,15 GET MPHONE
READ
RETURN
* Eof:

*PROGRAM UPSUPRD
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*UPDATES SUPPLIER INFORMATION IN SUPPLIER.DBF
@7,23 SAY MSUPNAME
@ 9,25 SAY MCONTACT
@ 14,16 SAY MSTREET
@ 15,14 SAY MCITY
@ 16,15 SAY MSTATE
@ 16,27 SAY MZIP
@ 19,15 SAY MPHONE
@ 9,25 GET MCONTACT
```

```
@ 14,16 GET MSTREET
@ 15,14 GET MCITY
@ 16,15 GET MSTATE
@ 16,27 GET MZIP
@ 19,15 GET MPHONE
READ
RETURN
* Eof:

*PROGRAM PRODOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR UPDATING PRODUCT CATALOGS
@ 4,27 SAY " A. ADD A NEW PRODUCT TO A CATALOG"
@ 6,27 SAY " B. UPDATE PRODUCT INFO"
@ 8,27 SAY " C. DELETE A PRODUCT FROM THE CATALOG"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof

* PROGRAM ENTPROD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
SELECT 2
CLEAR
DO WHILE .T.
  * Print the product entry form and read new values
  DO PRDFMASK
  DO ADDPRD
  * Check to see that all required values have been entered
  * And that there are no duplicate catalog #s for any one
  * supplier
  DO CHKPRD
  * If there has been an error in entry print the appropriate
  * message and allow it to be corrected
  IF ERROR
    @ 22,5 SAY ERRMESS
    WAIT
    @ 22,5
    @ 23,5
    ERRMESS=SPACE(60)
    LOOP
  ENDIF
  DO CONFIRMQ
  i=0
  DO CONFIRML
  DO CASE
    * Enter a new product in the product dbf
    CASE CHR(i) $ "Yy"
       APPEND BLANK
       REPLACE ITEMNO WITH MITEMNO
       REPLACE DESCRIPT WITH MDESCRIPT
```

```
        REPLACE SHIP_WT WITH MSHIPWT
        REPLACE UPRICE WITH MUPRICE
        REPLACE CAS_PRICE WITH MCASPRICE
        REPLACE SUPNAME WITH MSUPNAME
        DO MOREQ
        DO CONFIRML
        DO CASE
          * Rinitialize values for entrance of a new product
          CASE CHR(i) $"Yy"
            MITEMNO=0
            MDESCRIPT=SPACE(60)
            MSHIPWT=0
            MUPRICE=00.00
            MCASPRICE=000.00
            LOOP
          * Return to the previous menu
          CASE CHR(i) $ "Nn"
            EXIT
        ENDCASE
      * Allow correction of incorrect information
      CASE CHR(i) $ "Nn"
        LOOP
      * Return to previous menu
      CASE CHR(i) $ "Xx"
        EXIT
  ENDCASE
ENDDO
RETURN
*Eof

* PROGRAM UPDTPROD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
CLEAR
SUPMESS="UPDATE ITEMS IN CATALOG FOR SUPPLIER"
NOTEXIST="CATALOG # DOES NOT EXIST FOR THIS SUPPLIER--REENTER"
NOPRDS="NO PRODUCTS CURRENTLY IN THIS SUPPLIERS CATALOG"
DO WHILE .T.
  * Check to see that there are products for this supplier
  LOCATE FOR SUPNAME=MSUPNAME
  IF (.NOT. FOUND())
     @ 22,5 SAY NOPRDS
     WAIT
     @ 22,0
     @ 23,0
     EXIT
  ENDIF
  * Print product update menu and options
  DO BACKGRD3
  DO UPRDOPT
  DO CONFIRMM
  DO CASE
```

```
CASE CHR(i) $ "Aa"
  DO WHILE .T.
    i=0
    * Get the number of the item to update
    DO GETITNO
    * If the item does not exist for chosen supplier,
    * Print an error message and allow it to be reentered
    IF(.NOT. FOUND())
      @ 22,5 SAY NOTEXIST
      WAIT
      LOOP
    ENDIF
    * Print the product form and accept new values
    DO PRDFMASK
    DO CHANGCAT
    * Make sure values still exist in fields where values
    * Must exist- ignore the duplicate error message this time
    DO CHKPRD
    IF((ERROR) .AND. ((RTRIM(ERRMESS) <> MESS1)))
      @ 22,5 SAY ERRMESS
      WAIT
      @ 22,5
      @ 23,5
      LOOP
    ENDIF
    DO CONFIRMQ
    DO CONFIRML
    @ 22,5
    @ 23,5
    DO CASE
      * If everything is ok, update the product dbf
      CASE CHR(i) $ "Yy"
        DO MKPRDCHG
      * Allow corrections
      CASE CHR(i) $ "Nn"
        LOOP
      * Return to previous menu without making changes
      CASE CHR(i) $ "Xx"
        EXIT
    ENDCASE
    * Ask if user wants to make more changes
    @ 22,5 SAY "UPDATE ANOTHER PRODUCT? Y/N"
    DO CONFIRML
    IF CHR(i) $ "Yy"
      LOOP
    ELSE
      EXIT
    ENDIF
  ENDDO

  * Return to previous menu
  CASE CHR(i) $"Bb"
```

```
        EXIT

   ENDCASE
ENDDO
RETURN
*Eof


* PROGRAM PRDFMASK.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR PRODUCT ENTRY FORM
CLEAR
@ 0,0 TO 16,79 DOUBLE
@ 2,29 TO 4,50
@ 3,30 SAY "PRODUCT INFORMATION"
@ 7,5 SAY "SUPPLIER"
@ 7,40 SAY "CATALOG #:"
@ 9,5 SAY "ITEM DESCRIPTION"
@ 10,5 TO 12,67
@ 13,5 SAY "SHIPPING WT."
@ 13,20 SAY "UNIT PRICE"
@ 13,35 SAY "CASE PRICE"
RETURN
* Eof

* PROGRAM ADDPRD.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR ENTRY OF PRODUCT INFORMATION
@ 7,14 SAY MSUPNAME
@ 7,51 GET MITEMNO
@ 11,6 GET MDESCRIPT
@ 14,5 GET MSHIPWT
@ 14,20 GET MUPRICE
@ 14,35 GET MCASPRICE
READ
RETURN
* Eof

* Program chkprd
* Written by Gary Radke
* For Peoples Grocery Database
* Checks to see that newly entered catalog numbers
* are not duplicates of an existing catalog # for a given
* supplier.  Also checks to see that some value is
* entered in fields that require a value.
LOCATE FOR MSUPNAME=SUPNAME .AND. MITEMNO=ITEMNO
ERROR=.T.
DO CASE
  CASE (MITEMNO < 1)
    ERRMESS=MESS6
```

```
  CASE FOUND()
    ERRMESS=MESS1
  CASE (LEN(RTRIM(MDESCRIPT))<1)
    ERRMESS=MESS5
  CASE (MSHIPWT < 1)
    ERRMESS=MESS2
  CASE (MUPRICE < 0.01)
    ERRMESS=MESS3
  CASE (MCASPRICE < 0.01)
    ERRMESS=MESS4
  OTHERWISE
    ERROR=.F.
ENDCASE
RETURN
*Eof

*PROGRAM MOREQ.PRG
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
          @ 22,5 SAY "ENTER ANOTHER?  Y/N"
* Eof:

*PROGRAM UPRDOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR UPDATING THE MEMBER ORDER DATABASE
@ 4,27 SAY " A. ENTER AN ITEM #"
@ 6,27 SAY " B. RETURN TO THE PREVIOUS MENU"
@ 11,30 SAY "ENTER SELECTION A-B : :"
RETURN
*Eof

*PROGRAM GETITNO
*WRITTEN BY GARY RADKE
*FOR PEOPLES GROCERY DATABASE
CLEAR
@ 3,5 SAY "UPDATE CATALOG FOR PRODUCT #"
@ 3,37 GET MITEMNO
READ
USE PRODCAT
LOCATE FOR ITEMNO = MITEMNO .AND. SUPNAME=MSUPNAME
RETURN
* Eof:

* PROGRAM CHANGCAT.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR ENTRY OF PRODUCT INFORMATION
MSHIPWT=SHIP_WT
MDESCRIPT=DESCRIPT
MUPRICE=UPRICE
MCASPRICE=CAS_PRICE
```

```
@ 7,14 SAY MSUPNAME
@ 7,51 SAY MITEMNO
@ 11,6 GET MDESCRIPT
@ 14,5 GET MSHIPWT
@ 14,20 GET MUPRICE
@ 14,35 GET MCASPRICE
READ
RETURN
* Eof

* PROGRAM MKPRDCHG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
REPLACE DESCRIPT WITH MDESCRIPT
REPLACE SHIP_WT WITH MSHIPWT
REPLACE UPRICE WITH MUPRICE
REPLACE CAS_PRICE WITH MCASPRICE
RETURN
*Eof

*PROGRAM EXPOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR MAINTAINING THE COOPERATIVE EXPENSES
*DATABASE
@ 4,27 SAY " A. ENTER AN EXPENSE"
@ 6,27 SAY " B. BROWSE EXPENSE FILE"
@ 8,27 SAY " C. RETURN TO MAIN MENU"
@ 11,30 SAY "ENTER SELECTION A-C : :"
RETURN
*Eof

* PROGRAM MEMFMASK.PRG
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* TEMPLATE FOR EXPENSE INFORMATION FORM
CLEAR
@ 0,0 TO 14,79 DOUBLE
@ 2,31 TO 4,47
@ 3,33 SAY "EXPENSE FORM"
@ 7,9 SAY "DATE"
@ 7,50 SAY "AMOUNT"
@ 10,9 SAY "PAID TO"
@ 12,9 SAY "DESCRIPTION"
@ 13,9 SAY "PAID?"
RETURN
*Eof

*PROGRAM ADDEXP
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*ENTERS EXPENSE INFORMATION INTO EXPNEC.DBF
```

```
@ 7,14 GET MDATE
@ 7,60 GET MAMOUNT
@ 10,17 GET MPAIDTO
@ 12,21 GET MDESCRIPTI
READ
RETURN
* Eof:

* PROGRAM CHKEXP
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* CHECKS FOR DUPLICATE RECORDS AND MAKES SURE
* ENTRIES ARE MADE IN THE REQUIRED FIELDS
ERROR=.T.
LOCATE FOR DATE=MDATE .AND. PAID_TO=MPAIDTO .AND. AMOUNT=MAMOUNT;
.AND. DESCRIPTIO=MDESCRIPTI
DO CASE
  CASE FOUND()
    ERRMESS=MESS1
  CASE MAMOUNT < 0.01
    ERRMESS=MESS2
  CASE LEN(RTRIM(MPAIDTO)) < 1
    ERRMESS=MESS3
  CASE LEN(RTRIM(MDESCRIPTI)) < 1
    ERRMESS=MESS4
  OTHERWISE
    ERROR=.F.
ENDCASE
RETURN
* Eof

*PROGRAM BRSEXP
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*DISPLAYS EXPENSE INFORMATION IN EXPNEC.DBF
@ 7,14 SAY DATE
@ 7,60 SAY AMOUNT
@ 10,17 SAY PAID_TO
@ 12,21 SAY DESCRIPTIO
@ 13,15 SAY PAID
RETURN
* Eof:


*PROGRAM LABOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR GENERATING PRODUCT BREAKDOWN LABELS
@ 4,27 SAY " A. GENERATE PRODUCT BREAKDOWN LABELS FOR A"
@ 5,27 SAY "    SPECIFIED ORDER."
@ 8,27 SAY " B. RETURN TO MAIN MENU"
@ 11,30 SAY "ENTER SELECTION A-B : :"
```

```
RETURN
*Eof

* PROGRAM PRINTLBL
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* THIS PROGRAM PRINTS THE PRODUCT BREAK DOWN LABELS
* FOR A GIVEN ORDER

XIT=.F.
MMEMNO=0
MDATE=DATE()
MM4="ON "
GO1="PRINT LABELS FOR ORDER SUBMITTED TO "
LO1="ORDER DATE "
LO2="CATALOG # "
LO3="DESCRIPTION "
LO4="MEMBER # "
LO5="MEMBER NAME "
LO6="QUANTITY "
LO7="PRODUCT BREAK DOWN LABEL"
CENT=(79-LEN(LO7))/2
DO GETORD
* Let user return to main menu if order for label printing is not
* found
IF XIT
  RETURN
ENDIF
SELECT 5
USE MEMBER
INDEX ON MEM_NO TO MEMNO
SELECT 4
USE PRODCAT
SELECT 1
USE ITEM_ORD
JOIN WITH D TO ORDLFIL FOR ITEMNO = D->ITEMNO .AND. SUPNAME =
D->SUPNAME FIELDS ITEMNO, D->DESCRIPT,  QUANTITY, DATE, MEM_NO
INDEX ON DTOC(DATE)+STR(ITEMNO) TO LBLFIL
SELECT 1
USE ORDLFIL INDEX LBLFIL
LOCATE FOR DATE=MDATE
DO WHILE DATE=MDATE
   CLEAR
   MITEMNO=ITEMNO
   @ 1,CENT SAY LO7
   @ 2,5 SAY LO1
   @ 2,5+LEN(LO1) SAY DATE
   @ 3,5 SAY LO2
   @ 3,5+LEN(LO2) SAY ITEMNO
   @ 4,5 SAY LO3
   @ 4,5+LEN(LO3) SAY DESCRIPT
   @ 5,1 TO 5,79
```

```
    @ 7,5 SAY LO4
    @ 7,25 SAY LO5
    @ 7,45 SAY LO6
    ROWN=9
    DO WHILE ITEMNO=MITEMNO
      @ ROWN,5 SAY MEM_NO
      MMEMNO=MEM_NO
      SELECT 5
      USE MEMBER INDEX MEMNO
      SEEK MMEMNO
      @ ROWN,20 SAY RTRIM(FIRSTNAME)+" "+RTRIM(LASTNAME)
      SELECT 1
      @ ROWN,45 SAY QUANTITY
      ROWN=ROWN+1
      SKIP
      IF EOF()
        EXIT
      ENDIF
    ENDDO
    IF EOF()
      EXIT
    ENDIF
  ENDDO
WAIT
RELEASE ALL
CLOSE ALL
RETURN
*Eof


*PROGRAM BILLOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR PRODUCING ORDER BILLING REPORTS FOR EVERY
*MEMBER PARTICIPATING IN A GIVEN ORDER
@ 4,27 SAY " A. THIS MODULE PRODUCES A BILL ITEMIZING"
@ 5,27 SAY "    PRODUCTS PRICES, TOTAL MARKUP AND TAXES"
@ 6,27 SAY "    FOR EACH MEMBER PARTICIPATING IN A GIVEN"
@ 7,27 SAY "    ORDER."
@ 9,27 SAY " B. RETURN TO MAIN MENU"
@ 11,30 SAY "ENTER SELECTION A-B : :"
RETURN
*Eof

* PROGRAM GETORD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* LOCATES AN ORDER SUPPLIER AND DATE SO THE ORDER MAY BE
* ADJUSTED BEFORE BILLING OCCURS
GO2="Y/N"
GO3="NO OTHER ORDERS HAVE BEEN SUBMITTED.  DO YOU WANT TO LOOK"
GO4=" AT THE"
```

```
GO5="ORDER LIST AGAIN Y/N"
CLEAR
SELECT 1
USE ITEM_ORD INDEX STORORD
@ 3,5 SAY GO1
@ 3,5+LEN(GO1) SAY RTRIM(SUPNAME)
@ 3,5+LEN(GO1)+LEN(RTRIM(SUPNAME))+1 SAY MM4
@ 3,5+LEN(GO1)+LEN(RTRIM(SUPNAME))+1+LEN(MM4) SAY DATE
@ 4,5 SAY "Y/N ?"
DO CONFIRML
IF CHR(i) $ "Yy"
  MSUPNAME=SUPNAME
  MDATE=DATE
ELSE
  MSUPNAME=SUPNAME
  MDATE=DATE
  LOCATE FOR MDATE<>DATE .OR. MSUPNAME<>SUPNAME
  DO WHILE .T.
    IF (.NOT. EOF())
      @ 3,5 SAY GO1
      @ 3,5+LEN(GO1) SAY RTRIM(SUPNAME)
      @ 3,5+LEN(GO1)+LEN(RTRIM(SUPNAME))+1 SAY MM4
      @ 3,5+LEN(GO1)+LEN(RTRIM(SUPNAME))+1+LEN(MM4) SAY DATE
      @ 4,5 SAY "Y/N"
      DO CONFIRML
      DO CASE
         CASE CHR(i) $ "Yy"
           MSUPNAME=SUPNAME
           MDATE=DATE
           EXIT
         CASE CHR(i) $ "Nn"
           @ 3,5+LEN(GO1)
           CONTINUE
           MDATE=DATE
           MSUPNAME=SUPNAME
           LOOP
      ENDCASE
    ELSE
      @ 7,5 SAY GO3
      @ 7,5+LEN(GO3) SAY GO4
      @ 8,5 SAY GO5
      DO CONFIRML
      IF CHR(i) $ "Yy"
        GO TOP
        @ 7,5
        @ 8,5
        * LOCATE FOR MDATE<>DATE .OR. MSUPNAME<>SUPNAME
        LOOP
      ELSE
        XIT=.T.
        EXIT
      ENDIF
```

```
    ENDIF
  ENDDO
ENDIF
RETURN
*Eof

* PROGRAM ADJUST ORDER
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* THIS ROUTINE REMOVES REFUSED OR UNDELIVERED ITEMS
* FROM A GIVEN ORDER SO THAT A MEMBERS BILLS MAY BE PRINTED
* PROPERLY

AO1 ="ENTER THE CATALOG # OF AN UNDELIVERED OR REFUSED ITEM"
AO2 ="IF THERE ARE NO FURTHER UNDELIVERED OR REFUSED ITEMS"
AO3 ="TO BE ENTERED, TYPE 0000 "
AO4 ="REMOVE CATALOG # "
AO5 =" FROM THIS ORDER?  Y/N "
AO6 ="DELETE THIS ITEM FROM THE ORDER OF MEMBER # "
AO7 ="NO OTHER MEMBERS HAVE ORDERED "
AO8 ="? Y/N"
AO9 ="GIVEN ITEM # WAS NOT INCLUDED IN THIS ORDER"
RESPONSE=SPACE(4)
SELECT 6
USE MEMBER INDEX MEMNO
SELECT 1
DO WHILE .T.
  CLEAR
  @ 3,5 SAY AO1
  @ 4,5 SAY AO2
  @ 5,5 SAY AO3 GET RESPONSE
  READ
  IF RESPONSE  = "0000"
    EXIT
  ELSE
    LOCATE FOR ITEMNO=VAL(RESPONSE) .AND. DATE=MDATE .AND.
    SUPNAME=MSUPNAME
    IF (.NOT. FOUND())
      @ 7,5 SAY AO9
      WAIT
      @ 7,0
      LOOP
    ENDIF
    SELECT 4
    USE PRODCAT
    LOCATE FOR VAL(RESPONSE)=ITEMNO
    CLEAR
    @ 3,5 SAY AO4
    @ 3,5+LEN(AO4) SAY ITEMNO
    @ 4,5 SAY DESCRIPT
    @ 4,5+ LEN(RTRIM(DESCRIPT)) SAY AO5
    DO CONFIRML
```

```
    IF CHR(i) $ "Yy"
      SELECT 1
      LOCATE FOR VAL(RESPONSE)=ITEMNO .AND. MDATE=DATE
      THISMEM=MEM_NO
      CLEAR
      DO WHILE .T.
        IF .NOT. EOF()
          SELECT 6
          SEEK THISMEM
          MEMNAME=RTRIM(FIRSTNAME)+" "+RTRIM(LASTNAME)
          SELECT 1
          @ 3,5 SAY AO6
          CVAL=LEN(MEMNAME)
          @ 3,5+LEN(AO6) SAY LTRIM(STR(MEM_NO))
          @ 4,5 SAY MEMNAME
          @ 4,5 + CVAL SAY AO8
          DO CONFIRML
          IF CHR(i) $ "Yy"
            DELETE
            CONTINUE
            THISMEM=MEM_NO
          ELSE
            CONTINUE
            THISMEM=MEM_NO
          ENDIF
        ELSE
          SELECT 4
          @ 7,5 SAY AO7
          @ 7,5 + LEN(AO7) SAY DESCRIPT
          WAIT
          SELECT 1
          EXIT
        ENDIF
      ENDDO
    ENDIF
    RESPONSE="    "
  SELECT 1
  ENDDO
PACK
DISPLAY ALL
WAIT
SELECT 6
USE
SELECT 1
RETURN
END
*Eof

* PROGRAM PRINTBIL
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ROUTINE ACCEPTS INPUT OF SHIPPING COST, TAX, MEMBER EQUITY
```

```
* CHARGE AND SALES TAX AND CALCULATES THE AMOUNT A MEMBER OWES
* FOR THE ITEMS HE/SHE HAS RECEIVED IN THIS ORDER.
SET DECIMALS TO 2
BO1="ENTER THE SHIPPING COST IN PRICE PER POUND FOR THIS "
BO2="ORDER."
BO3="ENTER THE MEMBER EQUITY CHARGED FOR THIS ORDER AS PERCENT."
BO4="ENTER SALES TAX AS PERCENT"
BO5="ENTER MARKUP AS PERCENT"
BO6="ENTER ANY DISCOUNT AS PERCENT"
BO7="PRICE PER POUND SHIPPING"
BO8="TOTAL SHIPPING COST"
BO9="MARKUP"
B10="ORDER BILL FOR "
B11="MEMBER # "
B12="ORDER DATE "
B13="SUPPLIER "
B14="SUBTOTAL"
B15="TAX"
B16="DISCOUNT  -"
B17="EQUITY"
MESS1="SHIPPING COST MAY NOT BE LESS THAN 0"
MESS2="MEMBER EQUITY MAY NOT BE LESS THAN 0"
MESS3="MARKUP RATE MAY NOT BE LESS THAN 0"
MESS4="DISCOUNT RATE MAY NOT BE LESS THAN 0"
MESS5="TAX RATE MAY NOT BE LESS THAN 5.5%"
MESS6="PLEASE WAIT"
ERRMESS=SPACE(60)
ERR=.T.
ADJSTMNTS=0.00
SHIPRATE=0.00
EQUITYR=0.0
TAXRATE=5.5
MARKUPR=15.0
DISCOUNTR=0.0
SHIPCOST=0.00
EQUITYC=0.00
TAX=0.00
DISCOUNT=0.00
MARKUP=0.00
SUBTOT=0.00
UNITPCAS=0
WTPUNIT=0
SHIPWT=0
XIT=.F.
@ 3,5 SAY MESS6
SELECT 4
USE PRODCAT
SELECT 1
USE ITEM_ORD
JOIN WITH D TO ORDFIL FOR ITEMNO = D->ITEMNO .AND.;
SUPNAME = D->SUPNAME FIELDS D->DESCRIPT, ITEMNO, SUPNAME, QUANTITY,
MEM_NO,D->UPRICE, DATE, SHIP_WT, CAS_PRICE
```

```
INDEX ON DTOC(DATE)+STR(MEM_NO)+STR(ITEMNO)+SUPNAME TO BILFIL
SELECT 1
USE ORDFIL ALIAS ORDRINFO INDEX BILFIL
LOCATE FOR DATE=MDATE
@ 3,0
DO WHILE .T.
  DO BILADJ
  DO CONFIRMQ
  DO CONFIRML
  IF CHR(i) $ "Nn"
    LOOP
  ENDIF
  IF CHR(i) $ "Xx"
    XIT=.T.
  ENDIF
  @ 22,0
  @ 23,0
  DO CHKBADJ
  IF ERR
    @ 22,5 SAY ERRMESS
    WAIT
    ERRMESS = SPACE(60)
    @ 22,0
    @ 23,0
    LOOP
  ENDIF
  EXIT
ENDDO
IF XIT
  RETURN
ENDIF
DO WHILE .T.
  CLEAR
  MMEMNO=MEM_NO
  @ 5,5 SAY B10
  SELECT 5
  USE MEMBER
  LOCATE FOR MEM_NO=MMEMNO
  @ 5,5+LEN(B10) SAY RTRIM(FIRSTNAME)+" "+RTRIM(LASTNAME)
  SELECT ORDRINFO
  @ 5,60 SAY B11
  @ 5,60+LEN(B11) SAY MEM_NO
  @ 7,5 SAY B13
  @ 7,5+LEN(B13) SAY SUPNAME
  @ 7,55 SAY B12
  @ 7,55+LEN(B12)SAY DATE
  @ 9,0 TO 9,79 DOUBLE
  @ 10,3 SAY "CATALOG #"
  @ 10,15 SAY "DESCRIPTION"
  @ 10,35 SAY "QUANTITY ORDERED"
  @ 10,55 SAY "UNIT COST"
  @ 10,65 SAY "TOTAL PRICE"
```

```
@ 12,0 TO 12,79
ROWN=14
DO WHILE MMEMNO=MEM_NO .AND. MDATE=DATE

  @ ROWN,3 SAY ITEMNO
  @ ROWN,10 SAY DESCRIPT
  @ ROWN,35 SAY QUANTITY
  @ ROWN,55 SAY UPRICE
  PRICE=UPRICE*QUANTITY
  UNITPCAS=CAS_PRICE/UPRICE
  WTPUNIT=SHIP_WT/UNITPCAS
  SHIPWT=SHIPWT+(QUANTITY*WTPUNIT)
  SET DECIMALS TO 2
  UNITPCAS=0
  WTPUNIT=0
  @ ROWN,70 SAY PRICE PICTURE '@ ###.##'
  SUBTOT=SUBTOT + PRICE
  SKIP
  ROWN=ROWN+1
ENDDO
ROWN=ROWN+2
@ ROWN,50 SAY B14
@ ROWN,70 SAY SUBTOT PICTURE '@ ###.##'
EQUITYC=(EQUITYR/100)*SUBTOT
DISCOUNT=(DISCOUNTR/100)*SUBTOT
MARKUP=SUBTOT*(MARKUPR/100)
SUBTOT=SUBTOT+MARKUP
SHIPCOST=SHIPRATE*SHIPWT
ROWN=ROWN+1
@ ROWN,50 SAY B09
@ ROWN,70 SAY ROUND(MARKUP,2) PICTURE '@ ###.##'
ROWN=ROWN+1
@ ROWN,50 SAY B16
@ ROWN,70 SAY ROUND(DISCOUNT,2) PICTURE '@ ###.##'
ROWN=ROWN+1
SUBTOT=SUBTOT-DISCOUNT
TAX=(TAXRATE/100)*SUBTOT
SUBTOT=SUBTOT+TAX
@ ROWN ,50 SAY B15
@ ROWN,70 SAY ROUND(TAX,2) PICTURE '@ ###.##'
ROWN=ROWN+1
@ ROWN,50 SAY B17
@ ROWN,70 SAY ROUND(EQUITYC,2) PICTURE '@ ###.##'
ROWN=ROWN+1
SUBTOT=SUBTOT+EQUITYC
@ ROWN,50 SAY B08
@ ROWN,70 SAY ROUND(SHIPCOST,2) PICTURE '@ ###.##'
ROWN=ROWN+1
SUBTOT=SUBTOT+SHIPCOST
@ ROWN,65 TO ROWN,79 DOUBLE
@ ROWN+1,50 SAY "TOTAL COST     $"
@ ROWN+1,70 SAY ROUND(SUBTOT,2) PICTURE '@ ###.##'
```

```
  WAIT
  IF DATE=MDATE THEN
    I=INKEY()
        IF CHR(I) $ "X" THEN
          EXIT
        ELSE
          SUBTOT=0.00
          ADJSTMNTS=0.00
          TAX=0.00
          EQUITYC=0.00
          DISCOUNT=0.00
          MARKUP=0.00
          SHIPCOST=0.00
          SHIPWT=0
          LOOP
        ENDIF
  ELSE
    EXIT
  ENDIF
ENDDO
WAIT
RETURN
END
* Eof

* PROGRAM BILADJ
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* ALLOWS ENTRY OF MARKUP, TAXRATE, DISCOUNT RATE MEMBER EQUITY
* RATE, AND SHIPPING COST / POUND SO AN INDIVIDUAL MEMBERS BILL
* MAY BE ADJUSTED ACCORDINGLY

TITLE="ORDER ADJUSTMENTS"
@ 4,0 TO 20,79 DOUBLE
CENT=(79-LEN(TITLE))/2
@ 2,CENT SAY TITLE
@ 6,5 SAY BO1
@ 6,5+LEN(BO1)+LEN(BO2) SAY SHIPRATE
@ 6,5 +LEN(BO1) SAY BO2 GET SHIPRATE
@ B,5 + LEN(BO3) SAY EQUITYR
@ B,5 SAY BO3 GET EQUITYR
@ 11,5+LEN(BO4) SAY TAXRATE
@ 11,5 SAY BO4 GET TAXRATE
@ 13,5+LEN(BO5) SAY MARKUPR
@ 13,5 SAY BO5 GET MARKUPR
@ 15,5+LEN(BO6) SAY DISCOUNTR
@ 15,5 SAY BO6 GET DISCOUNTR
READ
RETURN
*Eof
```

```
* PROGRAM CHKBADJ
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* CHECKS TO SEE THAT ORDER ADJUSTMENT ENTRIES
* ARE NOT OUT OF BOUNDS

ERR=.T.
DO CASE
   CASE SHIPRATE < 0
     ERRMESS=MESS1
   CASE EQUITYR < 0
     ERRMESS=MESS2
   CASE MARKUPR < 0
     ERRMESS=MESS3
   CASE DISCOUNTR < 0
     ERRMESS=MESS4
   CASE TAXRATE < 5.5
     ERRMESS=MESS5
   OTHERWISE
     ERR=.F.
ENDCASE
RETURN
*Eof



*PROGRAM BACKGRD5
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*PRODUCES A GENERIC BACKGROUND FOR A MENU WITH FIVE OPTIONS.
CLEAR
@1,1 TO 16,77
@3,3 TO 14,75 DOUBLE
@ 2,2 SAY REPLICATE (CHR (176), 75)
@3,2 SAY CHR (176)
@ 4,2 SAY CHR (176)
@ 5,2 SAY CHR (176)
@ 6,2 SAY CHR (176)
@ 7,2 SAY CHR (176)
@ 8,2 SAY CHR (176)
@ 9,2 SAY CHR (176)
@ 10,2 SAY CHR (176)
@ 11,2 SAY CHR (176)
@ 12,2 SAY CHR (176)
@ 13,2 SAY CHR (176)
@ 14,2 SAY CHR (176)
@ 3,76 SAY CHR (176)
@ 4,76 SAY CHR (176)
@ 5,76 SAY CHR (176)
@ 6,76 SAY CHR (176)
```

```
@ 7,76 SAY CHR (176)
@ 8,76 SAY CHR (176)
@ 9,76 SAY CHR (176)
@ 10,76 SAY CHR (176)
@ 11,76 SAY CHR (176)
@ 12,76 SAY CHR (176)
@ 13,76 SAY CHR (176)
@ 14,76 SAY CHR (176)
@ 15,2 SAY REPLICATE (CHR (176), 75)
RETURN
*Eof

*PROGRAM CONSOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR GENERATING CONSIGNMENT SUMMARY REPORTS
@ 4,27 SAY " A. LIST PRODUCTS SOLD BUT MEMBER UNPAID"
@ 5,27 SAY "    FOR A GIVEN MEMBER"
@ 7,27 SAY " B. TOTAL ITEMS SOLD BUT MEMBERS UNPAID"
@ 8,27 SAY "    FOR ALL MEMBERS"
@ 10,27 SAY " C. TOTAL AMOUNT OWED FOR CONSIGNMENTS FOR"
@ 11,27 SAY "    SOLD AND UNSOLD ITEMS."
@ 13,27 SAY " D. RETURN TO MAIN MENU"
@ 15,30 SAY "ENTER SELECTION A-D : :"
RETURN
*Eof

* PROGRAM SOLDUPD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* GENERATES REPORT FOR CONSIGNMENTS WHICH HAVE BEEN SOLD THROUGH
* THE STORE FRONT BUT FOR WHICH THE CONSIGNING MEMBER HAS NOT
* BEEN PAID.

CLEAR
Y1=5
Y2=42
Y3=50
Y4=65
Y5=79
MEMESS="PREPARE CONSIGNMENT SUMMARY FOR THIS MEMBER? Y/N/X"
MESS1=" HAS NO SOLD BUT UNPAID CONSIGNMENTS"
HEADER1="SUMMARY OF CONSIGNMENTS SOLD BUT FOR WHICH MEMBER HAS "
HEADER2="NOT BEEN PAID"
HEADER3="MEMBER # "
HEADER4="MEMBER NAME "
COLUMN1="DESCRIPTION"
COLUMN2="UNITS"
COLUMN3="UNIT PRICE"
COLUMN4="CONSIGNMENT VALUE"
COLUMN5=""
TOT=0.00
```

```
GRANDTOT=0.00
XIT=.F.
SELECT 1
USE MEMBER INDEX MEMNO
SELECT 2
USE MEMBER_C
INDEX ON MEM_NO TO SOLDUORD
DO WHILE .T.
  SELECT 1
  DO GETMEM
  IF XIT
    RETURN
  ENDIF
  CLEAR
  NAME=RTRIM(FIRSTNAME)+" "+RTRIM(LASTNAME)
  SELECT 2
  LOCATE FOR (MEM_NO=A->MEM_NO) .AND. (ALL_SOLD) .AND. (.NOT.
MEMPAID)
  IF (.NOT. FOUND())
    @ 2,2 SAY NAME
    @ 2,2+LEN(RTRIM(NAME)) SAY MESS1
    WAIT
    @ 2,0
    LOOP
  ENDIF
  EXIT
ENDDO
LOCATE FOR MEM_NO=A->MEM_NO
@ 2,2 SAY HEADER1
@ 3,2 SAY HEADER2
@ 5,2 SAY HEADER3
@ 5,2+LEN(HEADER3) SAY MEM_NO
@ 5,30 SAY HEADER4
@ 5,30 + LEN(HEADER4) SAY NAME
@ 6,0 TO 6,79 DOUBLE
ROWN=7
DO PRINTCOL
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79
ROWN=ROWN+1
DO WHILE MEM_NO=A->MEM_NO
  IF (ALL_SOLD) .AND. (.NOT. MEMPAID) THEN
    TOT=TOT+(NO_OF_ITEM*CONSGRPRIC)
    GRANDTOT=GRANDTOT+TOT
    @ ROWN,5 SAY DESCRIPT
    @ ROWN,42 SAY NO_OF_ITEM
    @ ROWN,50 SAY CONSGRPRIC
    @ ROWN,65 SAY TOT
    TOT=0.00
    ROWN=ROWN+1
    SKIP
  ELSE
```

```
    SKIP
  ENDIF
  IF ROWN>55
    ROWN=4
    *    EJECT
    DO PRINTCOL
  ENDIF
ENDDO
@ ROWN,60 TO ROWN,79 DOUBLE
@ ROWN+1 ,2 SAY "TOTAL AMOUNT OWED THIS MEMBER FOR SOLD ITEMS"
@ ROWN+1, 65 SAY GRANDTOT
*EJECT
WAIT
CLOSE ALL
RETURN
END
* Eof

* PROGRAM ALLUSC
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* LISTS THE AMOUNT OWED TO EACH MEMBER FOR SOLD CONSIGNMENTS
* AND THE SUM OF THE AMOUNT OWED FOR SOLD CONSIGNMENTS

CLEAR
SELECT 1
USE MEMBER INDEX MEMNO
SELECT 2
USE MEMBER_C
INDEX ON MEM_NO TO SOLDORD
*DISPLAY ALL
*WAIT
HEADER1="SUMMARY OF SOLD, UPAID CONSIGNMENTS FOR ALL MEMBERS"
COLUMN1="MEMBER #"
COLUMN2="MEMBER NAME"
COLUMN3="AMOUNT OWED"
COLUMN4=""
COLUMN5=""
Y1=3
Y2=15
Y3=60
Y4=79
Y5=79
UPMEM=0
TOT=0.00
GRANDTOT=0.00
CENT=(79-LEN(HEADER1))/2
@ 2,CENT SAY HEADER1
ROWN=4
@ ROWN,0 TO ROWN,79
ROWN=5
DO PRINTCOL
```

```
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79 DOUBLE
ROWN=ROWN+1
MMEMNO=MEM_NO
DO WHILE .NOT. EOF()
  DO WHILE (MEM_NO=MMEMNO)
    IF (ALL_SOLD)  THEN
      IF .NOT. MEMPAID THEN
        TOT=TOT + (NO_OF_ITEM * CONSGRPRIC)
        SKIP
      ELSE
        SKIP
      ENDIF
    ELSE
      SKIP
    ENDIF
ENDDO
  IF TOT>0 THEN
    SELECT 1
    SEEK MMEMNO
    NAME=RTRIM(FIRSTNAME) +" "+RTRIM(LASTNAME)
    @ ROWN,5 SAY MEM_NO
    @ ROWN,15 SAY NAME
    @ ROWN,55 SAY TOT
    GRANDTOT=GRANDTOT+TOT
    TOT=0.00
    ROWN=ROWN+1
    SELECT 2
  ENDIF
  IF ROWN>55 THEN
    *    EJECT
    ROWN=3
    DO PRINTHDR
  ENDIF
MMEMNO=MEM_NO
ENDDO
@ ROWN,58 TO ROWN,70 DOUBLE
@  ROWN+1,0  SAY  "TOTAL  AMOUNT  OWED  TO  ALL  MEMBERS  FOR  SOLD
CONSIGNMENTS"
@ ROWN+1,55 SAY GRANDTOT
*EJECT
WAIT
CLOSE ALL
RETURN
*Eof

* PROGRAM ALLC
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* THIS PROGRAM GENERATES A REPORT THAT SUMMARIZES THE AMOUNT OWED
* TO MEMBERS FOR ALL SOLD AND UNSOLD CONSIGNMENTS
```

```
CLEAR
SELECT 1
USE MEMBER INDEX MEMNO
SELECT 2
USE MEMBER_C
INDEX ON MEM_NO TO SOLDORD
HEADER1="SUMMARY OF AMOUNT OWED TO MEMBERS FOR SOLD AND UNSOLD ì
CONSIGNMENTS"
COLUMN1="MEMBER #"
COLUMN2="MEMBER NAME"
COLUMN3="AMOUNT SOLD"
COLUMN4="AMOUNT UNSOLD"
COLUMN5=""
Y1=3
Y2=20
Y3=50
Y4=65
Y5=79
ROWN=5
ASOLD=0.00
ASUNSOLD=0.00
GTAS=0.00
GTAUS=0.00
@ 2,2 SAY HEADER1
@ 4,0 TO 4,79
DO PRINTCOL
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79 DOUBLE
ROWN=ROWN+1
MMEMNO=MEM_NO
DO WHILE .NOT. EOF()
  DO WHILE MEM_NO=MMEMNO
    DO CASE
      CASE (ALL_SOLD .AND. MEMPAID)
         SKIP
      CASE (ALL_SOLD .AND. (.NOT. MEMPAID))
         ASOLD=ASOLD+(NO_OF_ITEM*CONSGRPRIC)
         SKIP
      OTHERWISE
         ASUNSOLD=ASUNSOLD+(NO_OF_ITEM*CONSGRPRIC)
         SKIP
    ENDCASE
  ENDDO
  IF((ASOLD>0) .OR. (ASUNSOLD>0))
    GTAS=GTAS+ASOLD
    GTAUS=GTAUS+ASUNSOLD
    SELECT 1
    SEEK MMEMNO
    @ ROWN,5 SAY MEM_NO
    NAME=RTRIM(FIRSTNAME)+ " " + RTRIM(LASTNAME)
    @ ROWN,20 SAY NAME
    SELECT 2
```

```
    @ ROWN,45 SAY ASOLD
    @ ROWN,60 SAY ASUNSOLD
    ASOLD=0.00
    ASUNSOLD=0.00
    ROWN=ROWN+1
  ENDIF
  IF ROWN>55 THEN
    *    EJECT
    ROWN=5
    DO PRINTCOL
  ENDIF
MMEMNO=MEM_NO
ENDDO
ROWN=ROWN+1
@ ROWN+1,5 SAY "TOTAL AMOUNT OWED FOR SOLD "
@ ROWN+2,5 SAY "AND UNSOLD CONSIGNMENT ITEMS"
@ ROWN,45 TO ROWN,75 DOUBLE
@ ROWN+2,60 SAY GTAUS+GTAS
WAIT
*EJECT
CLOSE ALL
RETURN
*Eof

* PROGRAM PRINTCOL
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* PROGRAM PRINTS COLUMN HEADINGS FOR VARIOUS PROGRAMS
@ ROWN,Y1 SAY COLUMN1
@ ROWN,Y2 SAY COLUMN2
@ ROWN,Y3 SAY COLUMN3
@ ROWN,Y4 SAY COLUMN4
@ ROWN,Y5 SAY COLUMN5
RETURN
*Eof

*PROGRAM STOROPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR GENERATING A STORE ORDER REPORT
@ 4,27 SAY " A. LIST THE TOTAL ITEMIZED ORDER FOR A"
@ 5,27 SAY "    GIVEN DATE"
@ 7,27 SAY " B. RETURN TO MAIN MENU"
@ 11,22 SAY "ENTER SELECTION A-B AND TOUCH RETURN"
RETURN
*Eof

* PROGRAM COMBORD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* SUMMARIZES AN ORDER FOR A GIVEN DATE AND SUPPLIER
* PRINTS THE CATALOG #, ITEM DESCRIPTION, TOTAL QUANTITY
```

```
* WANTED, UNIT PRICE, TOTAL PRICE, AND TOTAL VALUE OF THE ORDER
CLEAR
HEADER1="ORDER SUMMARY FOR "
GO1="PREPARE STORE ORDER FOR ORDER SUBMITTED TO "
MM4="ON "
COLUMN1="CATALOG #"
COLUMN2="DESCRIPTION"
COLUMN3="# CASES"
COLUMN4="CASE PRICE"
COLUMN5="TOTAL COST"
Y1=0
Y2=12
Y3=47
Y4=56
Y5=70
TDATE=DATE()
TOTQUANT=0
TOTPRIC=0
SELECT 3
USE PRODCAT INDEX PRODORD
GRANDTOT=0.00
SELECT 1
USE ITEM_ORD
INDEX ON DTOC(DATE) + SUPNAME + STR(ITEMNO,4) TO STORORD
DO GETORD
IF XIT
  XIT=.F.
  RETURN
ENDIF
TDATE=DATE
MSUPNAME=SUPNAME
CLEAR
CENT=(79-(LEN(HEADER1)+LEN(RTRIM(SUPNAME))+LEN(DTOC(DATE))))/2
@ 3,CENT SAY HEADER1
@ 3,CENT+LEN(HEADER1)+1 SAY SUPNAME
@ 3,CENT+LEN(HEADER1)+LEN(RTRIM(SUPNAME))+2 SAY DATE
ROWN=4
@ ROWN,0 TO ROWN,79
ROWN=ROWN+1
DO PRINTCOL
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79 DOUBLE
ROWN=ROWN+1
DO WHILE (DATE=TDATE) .AND. (SUPNAME=MSUPNAME)
   MITEMNO=ITEMNO
   SELECT 3
   SEEK MITEMNO
   MDESCRIPT=DESCRIPT
   MCASPRIC=CAS_PRICE
   MWEIGHT=SHIP_WT
   MPRICE=UPRICE
   SELECT 1
```

```
   DO WHILE ITEMNO=MITEMNO
     TOTQUANT=TOTQUANT+QUANTITY
     SKIP
   ENDDO
   TOTPRICE=TOTQUANT*C->UPRICE
   NUMCASE=TOTPRICE/C->CAS_PRICE
   @ ROWN,Y1 SAY MITEMNO
   @ ROWN,Y2 SAY C->DESCRIPT
   @ ROWN,Y3 SAY NUMCASE PICTURE '@ ###'
   @ ROWN,Y4 SAY C->CAS_PRICE PICTURE '@ ###.##'
   @ ROWN,Y5 SAY TOTPRICE PICTURE '@ ###.##'
   GRANDTOT=GRANDTOT+TOTPRICE
   TOTPRICE=0.00
   ROWN=ROWN+1
   IF ROWN> 65 THEN
      *      EJECT
      ROWN=4
      DO PRINTCOL
   ENDIF
ENDDO
@ ROWN,Y5-5 TO ROWN,79 DOUBLE
ROWN=ROWN+1
@ ROWN,Y2 SAY "TOTAL ORDER COST"
@ ROWN,Y4 SAY "$"
@ ROWN,Y5-2 SAY GRANDTOT PICTURE '@ #,###.##'
WAIT
CLOSE ALL
RETURN
* Eof

*PROGRAM FINOPT
*WRITTEN BY GARY RADKE
*FOR THE PEOPLES GROCERY DATABASE
*LISTS THE OPTIONS FOR GENERATING FINANCE SUMMARY REPORTS
@ 4,22 SAY " A. LIST ITEMIZED EXPENSES FOR A GIVEN MONTH"
@ 6,22 SAY " B. LIST TOTAL MONTHLY EXPENSES & GROSS SALES"
@ 7,22 SAY "    FOR A GIVEN MONTH"
@ 9,22 SAY " C. LIST OUTSTANDING DEBTS INCLUDING AMOUNTS"
@ 10,22 SAY "    OWED FOR SOLD AND UNSOLD CONSIGNED ITEMS."
@ 12,22 SAY " D. RETURN TO MAIN MENU"
@ 15,30 SAY "ENTER SELECTION A-D : :"
RETURN
*Eof

* PROGRAM LISTEX
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* THE PROGRAM GENERATES A REPORT THAT LISTS EXPENSES FOR THE
* IN A GIVEN MONTH
CLEAR
HEADER1="SUMMARY OF STORE EXPENSES FOR THE MONTH OF"
COLUMN1="DATE"
```

```
COLUMN2="PAID TO"
COLUMN3="DESCRIPTION"
COLUMN4="AMOUNT"
COLUMN5=""
Y1=5
Y2=16
Y3=38
Y4=72
Y5=79
TDATE=DATE()
TOT=0.00
ROWN=5
DO GETPERIO
CLEAR
@ ROWN,5 SAY HEADER1
@ ROWN,5+LEN(HEADER1)+1 SAY CMONTH(TDATE)
@ ROWN,5+LEN(HEADER1)+LEN(CMONTH(TDATE))+1 SAY YEAR(TDATE)
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79
ROWN=ROWN+1
DO PRINTCOL
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79 DOUBLE
SELECT 1
USE EXPENC
INDEX ON DATE TO EXPORD
LOCATE FOR MONTH(DATE)=MONTH(TDATE)
ROWN=ROWN+1
DO WHILE MONTH(DATE)=MONTH(TDATE)
  @ ROWN,Y1 SAY DATE
  @ ROWN,Y2 SAY PAID_TO
  @ ROWN,Y3 SAY DESCRIPTIO
  @ ROWN,Y4 SAY AMOUNT
  TOT=TOT+AMOUNT
  ROWN=ROWN+1
  SKIP
  IF ROWN>60
    *   EJECT
    ROWN=4
    DO PRINTCOL
  ENDIF
ENDDO
@ ROWN,Y4 TO ROWN,79 DOUBLE
@ ROWN+1,15 SAY "TOTAL MONTHLY EXPENSES"
@ ROWN+1,Y4-5 SAY '$'
@ ROWN+1,Y4-1 SAY TOT PICTURE '@ #,###.##'
*EJECT
WAIT
CLOSE ALL
RETURN
END
*Eof
```

```
* PROGRAM EXPC_SAL
* WRITTEN BY GARY RADKE
* PROGRAM ITEMIZES MONTHLY EXPENSES AND GROSS SALES FOR A
* GIVEN MONTH
CLEAR
TOTSALE=0.00
TOTEXP=0.00
TDATE=DATE()
TITEM=0
HEADER1="MONTHLY GROSS SALES / EXPENSE REPORT"
HEADER2="FOR THE MONTH OF"
SALELBL="TOTAL GROSS SALES FOR THIS MONTH"
EXPLBL="TOTAL EXPENSES FOR THIS MONTH"
DO GETPERIOD
CLEAR
SELECT 2
USE PRODCAT
INDEX ON ITEMNO TO PRODORD
SELECT 1
USE ITEM_ORD
INDEX ON DATE TO EXPFIL
LOCATE FOR MONTH(DATE)=MONTH(TDATE) .AND. YEAR(DATE)=YEAR(TDATE)
TITEM=ITEMNO
DO WHILE (MONTH(DATE)=MONTH(TDATE)) .AND. (YEAR(DATE)=YEAR(TDATE))
  SELECT 2
  SEEK TITEM
  TOTSALE=TOTSALE+(A->QUANTITY * UPRICE)
  SELECT 1
  SKIP
  TITEM=ITEMNO
ENDDO
SELECT 3
USE EXPENC
INDEX ON DATE TO EXPORD
LOCATE FOR MONTH(DATE)=MONTH(TDATE) .AND. YEAR(DATE)=YEAR(TDATE)
DO WHILE MONTH(DATE)=MONTH(TDATE) .AND. YEAR(DATE)=YEAR(TDATE)
  TOTEXP=TOTEXP+AMOUNT
  SKIP
ENDDO
CENT=(79-LEN(HEADER1))/2
@ 3,CENT SAY HEADER1
@ 5,5 SAY HEADER2
@ 5,5+LEN(HEADER2)+1 SAY CMONTH(TDATE)
@ 5,7+LEN(HEADER2)+LEN(CMONTH(TDATE)) SAY YEAR(TDATE)
@ 7,0 TO 7,79
@ 10,5 SAY SALELBL
@ 10,15+LEN(SALELBL) SAY '$'
@ 10,15 + LEN(SALELBL)+5 SAY TOTSALE PICTURE '@ #,###.##'
@ 12,5 SAY EXPLBL
@ 12,15+LEN(SALELBL) SAY '$'
@ 12,15 + LEN(SALELBL)+5 SAY TOTEXP PICTURE '@ #,###.##'
```

```
@ 15,0 TO 15,79 DOUBLE
WAIT
*EJECT
CLOSE ALL
RETURN
* Eof

* PROGRAM CONEXP
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* PROGRAM PRODUCES A REPORT THAT SUMMARIZES ALL OUTSTANDING
* DEBTS INCLUDING UNPAID CONSIGNMENTS

CLEAR
HEADER1="REPORT OF ALL OUTSTANDING DEBTS"
HEADER2="UNPAID BILLS"
HEADER3="UNPAID CONSINGMENTS"
HEADER4="TOTAL UNPAID BILLS"
HEADER5="TOTAL UNPAID CONSIGNMENTS"
COLUMN1="PAID TO"
COLUMN2="DESCRIPTION"
COLUMN3="AMOUNT"
COLUMN4=""
COLUMN5=""
Y1=3
Y2=28
Y3=68
Y4=79
Y5=79
NAME=SPACE(25)
UPBILLS=0.00
TOTOWED=0.00
OWEDTM=0.00
TEMPMEM=0
ROWN=3
CENT=(79-LEN(HEADER1))/2
@ ROWN,CENT SAY HEADER1
ROWN=ROWN+2
@ ROWN,0 SAY HEADER2
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79
ROWN=ROWN+1
DO PRINTCOL
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79 DOUBLE
ROWN=ROWN+1
SELECT 3
USE MEMBER INDEX MEMNO
SELECT 1
USE EXPENC
DO WHILE .NOT. EOF()
  IF .NOT. PAID
```

```
    @ ROWN,Y1 SAY PAID_TO
    @ ROWN,Y2 SAY DESCRIPTIO
    @ ROWN,Y3 SAY AMOUNT
    TOTOWED=TOTOWED+AMOUNT
  ENDIF
  ROWN=ROWN+1
  SKIP
ENDDO
UPBILLS=TOTOWED
@ ROWN,60 TO ROWN,79
ROWN=ROWN+1
@ ROWN,35 SAY HEADER4
@ ROWN,Y3-5 SAY '$'
@ ROWN,Y3 SAY UPBILLS PICTURE '@ #,###.##'
WAIT
CLEAR
ROWN=3
COLUMN1="NAME"
COLUMN2="CONSIGNMENT AMOUNT"
COLUMN3=""
COLUMN4=""
COLUMN5=""
Y2=60
Y3=79
@ ROWN,0 TO ROWN,79
ROWN=ROWN+2
@ ROWN,0 SAY HEADER3
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79
ROWN=ROWN+1
DO PRINTCOL
ROWN=ROWN+1
@ ROWN,0 TO ROWN,79 DOUBLE
ROWN=ROWN+1
SELECT 2
USE MEMBER_C
INDEX ON MEM_NO TO CONORD
DO WHILE .NOT. EOF()
  TEMPMEM=MEM_NO
  SELECT 3
  SEEK TEMPMEM
  NAME=RTRIM(FIRSTNAME) + " "+ RTRIM(LASTNAME)
  SELECT 2
  DO WHILE MEM_NO=TEMPMEM
    IF (.NOT. MEMPAID)
      OWEDTM=OWEDTM+(NO_OF_ITEM*CONSGRPRIC)
      TOTOWED=TOTOWED+(NO_OF_ITEM*CONSGRPRIC)
    ENDIF
    SKIP
  ENDDO
  @ ROWN,Y1 SAY NAME
  @ ROWN,Y2 SAY OWEDTM PICTURE '@ #,###.##'
```

```
  OWEDTM=0.00
  ROWN=ROWN+1
  IF ROWN>55 THEN
    *   EJECT
    ROWN=4
  ENDIF
ENDDO
@ ROWN,60 TO ROWN,79
ROWN=ROWN+1
@ ROWN,25 SAY HEADER5
@ ROWN,Y2-5 SAY '$'
@ ROWN,Y2 SAY TOTOWED-UPBILLS PICTURE '@ #,###.##'
@ ROWN+2,60 TO ROWN+2,79 DOUBLE
@ ROWN+3,25 SAY "TOTAL OUTSTANDING DEBT"
@ ROWN+3,Y2-5 SAY '$'
@ ROWN+3,Y2 SAY TOTOWED PICTURE '@ #,###.##'
*EJECT
WAIT
CLOSE ALL
RELEASE ALL
RETURN
* Eof

* PROGRAM GETPERIOD
* WRITTEN BY GARY RADKE
* FOR PEOPLES GROCERY DATABASE
* LETS USER ENTER THE MONTH AND YEAR FOR WHICH THEY WANT TO
* SUMMARIZE COOP FINANCES

CLEAR
ERMESS="CAN'T LIST EXPENSES FOR A FUTURE MONTH. PLEASE REENTER"
MESS1="ENTER THE MONTH AND YEAR YOU WANT SUMMARIZED."
MESS2="DATE "
DO WHILE .T.
  @ 3,5 SAY MESS1
  @ 5,5 SAY MESS2
  @ 5,LEN(MESS2)+5 SAY DATE()
  @ 5,LEN(MESS2)+5 GET TDATE
  READ
  IF (TDATE <= DATE()) THEN
    EXIT
  ELSE
    CLEAR
    @ 3,5 SAY ERMESS
    WAIT
    CLEAR
    LOOP
  ENDIF
ENDDO
RETURN
*Eof
```

```
$COMMAND
PG
SET MESSAGE OFF
OPEN PEOPLES
SET ERROR MESSAGE OFF
SET COLOR BACKGRND BLACK
SET COLOR FOREGRND GRAY
SET BELL OFF
SET VAR PICK1 INT
LABEL STARTAPP
  NEWPAGE
  CHOOSE PICK1 FROM Main IN PG.APX
  IF PICK1 EQ 0 THEN
    GOTO ENDAPP
  ENDIF
  IF PICK1 EQ 1 THEN
    SET VAR PICK2 INT
    SET VAR LEVEL2 INT
    SET VAR LEVEL2 TO 0
    WHILE LEVEL2 EQ 0  THEN
      NEWPAGE
      CHOOSE PICK2 FROM ENTUPD IN PG.APX
      IF PICK2 EQ 0 THEN
        BREAK
      ENDIF
      IF PICK2 EQ 1 THEN
        SET VAR PICK3 INT
        SET VAR LEVEL3 INT
        SET VAR LEVEL3 TO 0
        WHILE LEVEL3 EQ 0  THEN
          NEWPAGE
          CHOOSE PICK3 FROM MEMI IN PG.APX
          IF PICK3 EQ 0 THEN
            BREAK
          ENDIF
          IF PICK3 EQ 1 THEN
            ENTER  MEMINF
          ENDIF
          IF PICK3 EQ 2 THEN
            RUN STRULOF  IN PG.APX
            SET VARIABLE WHVAL1      TO  TEXT
            FILLIN WHVAL1       USING "ENTER # OF MEMBER TO UPDATE
            "
            EDIT USING  MEMINF    +
              SORTED BY  mem_no=A +
              WHERE mem_no   EQ        .WHVAL1
            CLEAR WHVAL1
            RUN STRULON  IN PG.APX
          ENDIF
```

```
    IF PICK3 EQ 3 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 2 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM SUPINF IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK3 EQ 1 THEN
      ENTER  SUPPLIER
    ENDIF
    IF PICK3 EQ 2 THEN
      RUN SUPRULOF IN PG.APX
      SET VARIABLE WHVAL1      TO  TEXT
      FILLIN WHVAL1       USING "ENTER NAME OF SUPPLIER TO
      UPDATE "
      EDIT USING  SUPPLIER  +
        SORTED BY  supname=A +
        WHERE supname  EQ       .WHVAL1
      CLEAR WHVAL1
      RUN SUPRULON IN PG.APX
    ENDIF
    IF PICK3 EQ 3 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 3 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM PRDCAT IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK3 EQ 1 THEN
      ENTER  PRODUCT
    ENDIF
    IF PICK3 EQ 2 THEN
```

```
      RUN STPRULOF IN PG.APX
      SET VARIABLE WHVAL1       TO  TEXT
      FILLIN WHVAL1       USING "PLEASE ENTER SUPPLIER NAME
      "
      SET VARIABLE WHVAL2       TO  `┌Æ┌Ä┌è┌
      FILLIN WHVAL2       USING "ENTER CATALOG # OF ITEM TO
      UPDATE "
      EDIT USING  PRODUCT   +
        SORTED BY  supname=A itemno=A +
        WHERE supname  EQ        .WHVAL1        +
        AND itemno   EQ        .WHVAL2
      CLEAR WHVAL1
      CLEAR WHVAL2
      RUN STPRULON IN PG.APX
    ENDIF
    IF PICK3 EQ 3 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 4 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM MEMORD IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK3 EQ 1 THEN
      ENTER  ITEMORD
      RUN DDUPITOR IN PG.APX
    ENDIF
    IF PICK3 EQ 2 THEN
      SET VARIABLE WHVAL1       TO  TEXT
      FILLIN WHVAL1       USING "UPDATE ORDER FOR WHICH
      MEMBER # ? "
      SET VARIABLE WHVAL2       TO  DATE
      FILLIN WHVAL2       USING "ENTER DATE OF ORDER TO UPDATE
     "
      EDIT  +
        item_no   date       quantity  supname   +
        FROM  item_ord  +
        SORTED BY  date=D +
        WHERE mem_no   EQ        .WHVAL1        +
        AND date     EQ        .WHVAL2
      CLEAR WHVAL1
      CLEAR WHVAL2
      RUN D2DUPITO IN PG.APX
```

```
    ENDIF
    IF PICK3 EQ 3 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 5 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM CONSGMTS IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK3 EQ 1 THEN
      ENTER  CONSIGN
      RUN CONDUPD  IN PG.APX
    ENDIF
    IF PICK3 EQ 2 THEN
      SET VARIABLE WHVAL1       TO  TEXT
      FILLIN WHVAL1       USING "UPDATE CONSIGNMENTS FOR WHICH
     MEMBER # ? "
      EDIT  +
        date       descript  consgrpr  no_of_it  mempaid
   all_sold  +
        FROM  member_c  +
        SORTED BY  date=D +
        WHERE mem_no   EQ       .WHVAL1
      CLEAR WHVAL1
      RUN CONMDUPD IN PG.APX
    ENDIF
    IF PICK3 EQ 3 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 6 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM EXPENSES IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
```

```
        IF PICK3 EQ 1 THEN
          ENTER  EXPENSES
          RUN EXPDUPD  IN PG.APX
        ENDIF
        IF PICK3 EQ 2 THEN
          EDIT  +
            date      amount    paid_to   descript  paid      +
            FROM  expenc    +
            SORTED BY  date=D
          RUN EXPMDUPD IN PG.APX
        ENDIF
        IF PICK3 EQ 3 THEN
          BREAK
        ENDIF
      ENDWHILE
      CLEAR LEVEL3
      CLEAR PICK3
    ENDIF
  ENDWHILE
  CLEAR LEVEL2
  CLEAR PICK2
  GOTO STARTAPP
ENDIF
IF PICK1 EQ 2 THEN
  SET VAR PICK2 INT
  SET VAR LEVEL2 INT
  SET VAR LEVEL2 TO 0
  WHILE LEVEL2 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK2 FROM REPTMAIN IN PG.APX
    IF PICK2 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK2 EQ 1 THEN
      SET VAR PICK3 INT
      SET VAR LEVEL3 INT
      SET VAR LEVEL3 TO 0
      WHILE LEVEL3 EQ 0  THEN
        NEWPAGE
        CHOOSE PICK3 FROM PRTL IN PG.APX
        IF PICK3 EQ 0 THEN
          BREAK
        ENDIF
        IF PICK3 EQ 1 THEN
          RUN PRTLBL   IN PG.APX
        ENDIF
        IF PICK3 EQ 2 THEN
          BREAK
        ENDIF
      ENDWHILE
      CLEAR LEVEL3
      CLEAR PICK3
```

```
ENDIF
IF PICK2 EQ 2 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM PRBIL IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK3 EQ 1 THEN
      RUN PB        IN PG.APX
    ENDIF
    IF PICK3 EQ 2 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 3 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
    NEWPAGE
    CHOOSE PICK3 FROM CONSGMT IN PG.APX
    IF PICK3 EQ 0 THEN
      BREAK
    ENDIF
    IF PICK3 EQ 1 THEN
      RUN CONUPD    IN PG.APX
    ENDIF
    IF PICK3 EQ 2 THEN
      RUN AUPD      IN PG.APX
    ENDIF
    IF PICK3 EQ 3 THEN
      RUN AUNPD     IN PG.APX
    ENDIF
    IF PICK3 EQ 4 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL3
  CLEAR PICK3
ENDIF
IF PICK2 EQ 4 THEN
  SET VAR PICK3 INT
  SET VAR LEVEL3 INT
  SET VAR LEVEL3 TO 0
  WHILE LEVEL3 EQ 0  THEN
```

```
        NEWPAGE
        CHOOSE PICK3 FROM COMBO IN PG.APX
        IF PICK3 EQ 0 THEN
          BREAK
        ENDIF
        IF PICK3 EQ 1 THEN
          RUN COMBORD  IN PG.APX
        ENDIF
        IF PICK3 EQ 2 THEN
          BREAK
        ENDIF
      ENDWHILE
      CLEAR LEVEL3
      CLEAR PICK3
    ENDIF
    IF PICK2 EQ 5 THEN
      SET VAR PICK3 INT
      SET VAR LEVEL3 INT
      SET VAR LEVEL3 TO 0
      WHILE LEVEL3 EQ 0  THEN
        NEWPAGE
        CHOOSE PICK3 FROM EXPENSE IN PG.APX
        IF PICK3 EQ 0 THEN
          BREAK
        ENDIF
        IF PICK3 EQ 1 THEN
          RUN LISTEX   IN PG.APX
        ENDIF
        IF PICK3 EQ 2 THEN
          RUN OD       IN PG.APX
        ENDIF
        IF PICK3 EQ 3 THEN
          RUN EXCON    IN PG.APX
        ENDIF
        IF PICK3 EQ 4 THEN
          BREAK
        ENDIF
      ENDWHILE
      CLEAR LEVEL3
      CLEAR PICK3
    ENDIF
    IF PICK2 EQ 6 THEN
      BREAK
    ENDIF
  ENDWHILE
  CLEAR LEVEL2
  CLEAR PICK2
  GOTO STARTAPP
ENDIF
IF PICK1 EQ 3 THEN
  GOTO ENDAPP
ENDIF
```

```
  GOTO STARTAPP
LABEL ENDAPP
CLEAR PICK1
RETURN
$MENU
Main
COLUMN PEOPLES GROCERY DATABASE
ENTER/UPDATE INFORMATION
REPORTS
EXIT
$MENU
ENTUPD
COLUMN ENTER/UPDATE INFORMATION
MEMBER SHIP INFORMATION
SUPPLIER INFORMATION
PRODUCT CATALOG
MEMBER ORDERS
CONSIGNMENTS
EXPENSES
$MENU
MEM1
COLUMN MEMBER SHIP INFORMATION
ENTER A NEW MEMBER
UPDATE INFORMATION FOR AN EXISTING MEMBER
EXIT
$MENU
SUPINF
COLUMN SUPPLIER INFORMATION
ADD A NEW SUPPLIER
UPDATE AN EXISTING SUPPLIER
RETURN TO PREVIOUS MENU
$MENU
PRDCAT
COLUMN PRODUCT CATALOG
ENTER A NEW PRODUCT
UPDATE PRODUCT INFORMATION
EXIT
$MENU
MEMORD
COLUMN MEMBER ORDERS
ENTER AN ITEM ORDER
UPDATE AN ITEM ORDER
EXIT
$MENU
CONSGMTS
COLUMN ENTER/UPDATE CONSIGNMENTS
ENTER A CONSIGNMENT
UPDATE CONSIGNMENTS
EXIT
$MENU
EXPENSES
COLUMN ENTER/UPDATE EXPENSES
```

```
ENTER AN EXPENSE
UPDATE AN EXPENSE
EXIT
$MENU
REPTMAIN
COLUMN PEOPLES GROCERY REPORTS
PRODUCT DISTRIBUTION LABELS
PRINT MEMBER ORDER BILLS
CONSIGNMENT SUMMARY
COMBINED ORDER
STORE EXPENSE SUMMARY
EXIT
$MENU
CONSGMT
COLUMN CONSIGNMENT SUMMARY
LIST PRODUCTS SOLD BUT MEMBER UNPAID FOR A GIVEN MEMBER
LIST TOTAL ITEMS SOLD BUT MEMBERS UNPAID FOR ALL MEMBERS
LIST TOTAL AMOUNT OWED FOR SOLD AND UNSOLD CONSIGNMENTS
EXIT
$MENU
COMBO
COLUMN COMBINED ORDER REPORT
ITEMIZE TOTAL ORDER FOR A GIVEN DATE AND SUPPLER
EXIT
$MENU
EXPENSE
COLUMN EXPENSE REPORT MENU
LIST ITEMIZED EXPENSES FOR A GIVEN MONTH
LIST TOTAL EXPENSES AND GROSS SALES FOR A GIVEN MONTH
LIST OUTSTANDING DEBTS INCLUDING AMOUNT OWED FOR CONSIGNMENT
EXIT
$MENU
PRTL
COLUMN PRODUCT DISTRIBUTION LABELS
PRINT PRODUCT DISTRIBUTION LABELS FOR A SPECIFIED ORDER
EXIT
$MENU
PRBIL
COLUMN ORDER BILLS
PRINT ITEMIZED BILLS FOR EACH MEMBER IN A SPECIFIED ORDER
EXIT
$COMMAND
STRULOF
SET RULES OFF
RETURN

$COMMAND
STRULON
SET RULES ON
RETURN

$COMMAND
```

```
SUPRULOF
SET RULES OFF
RETURN

$COMMAND
SUPRULON
SET RULES ON
RETURN

$COMMAND
STPRULOF
SET RULES OFF
RETURN

$COMMAND
STPRULON
SET RULES ON
RETURN

$COMMAND
DDUPITOR
delete duplicates from item_ord

$COMMAND
D2DUPITO
delete duplicates from item_ord

$COMMAND
CONDUPD
delete duplicates from member_c
return


$COMMAND
CONMDUPD
delete duplicates from member_c
return

$COMMAND
EXPDUPD
delete duplicates from expenc
return

$COMMAND
EXPMDUPD
delete duplicates from expenc
return

$COMMAND
CONUPD
*(THIS MODULE ITEMIZES CONSIGNMENTS WHICH HAVE BEEN SOLD BUT FOR
WHICH THE)
```

```
*(CONSIGNER HAS NOT BEEN PAID)
*(WRITTEN BY GARY RADKE)
*(FOR PEOPLES GROCERY DATABASE)

cls
set messages off
set error messages off
set var qpaid to n        *(conditional comparison values)
set var qsold to y        *(                              )
set var tmem integer      *(member #)
set var answer to text      *(generic variable used to evaluate
queries)
set var fname to text     *(full name of member)
set var rown to 8         *(row position designator)
set var ttotal currency   *(total value of sold unpaid consignments)
set var tquant integer      *(number of a given item placed on
consignment)
set var tuprice currency   *(unit price of a given item placed on
consignment)
set var header to "SUMMARY OF SOLD BUT UNPAID CONSIGNMENTS"
set var total to "TOTAL SOLD BUT UNPAID CONSIGNMENTS"
set var ttotal = 0
set var cont to "press any key to continue"
label entmem
*(get number of member to summarize)
run entmem in consmod
*(if no number is entered, give user a chance to quit otherwise
reenter)
if tmem fails then
  if answer ne x then
    goto entmem
  else
    goto quit
  endif
endif
cls
write "PLEASE WAIT"
*(create a temporary file containing only those records of sold
unpaid)
*(consignments for all members )
project upcons from member_c using mem_no where mempaid eq .qpaid
and +
all_sold eq .qsold
delete duplicates from upcons
*(check to see if the chosen member # exists in the temporary file,
if not)
*(report the error and show which member #s do exist and allow
reentry of #)
set pointer #1 pl for upcons where mem_no eq .tmem
if pl ne 0 then
   run nocons in consmod
   goto entmem
```

```
endif
cls
remove upcons
*(create a temporary file containing only those records of sold,
unpaid)
*(consignments for the chosen member)
project consold from member_c using all sorted by date where mem_no
eq +.tmem and mempaid eq .qpaid and all_sold eq .qsold
set var tfirst to firstnam in member where mem_no eq .tmem
set var tlast to lastname in member where mem_no eq .tmem
set var fname to (.tfirst & .tlast)
run conshdr in consmod
set pointer #1 p1 for consold where mem_no eq .tmem
while p1 eq 0 then
  set var tdescrip to descript in #1
  set var tquant to no_of_it in #1
  set var tuprice to consgrpr in #1
  set var tvalue = .tquant * .tuprice
  set var ttotal = .ttotal + .tvalue
  write .tdescrip at .rown,2
  write .tquant at .rown,35
  write .tuprice at .rown,45
  write .tvalue at .rown,57
  next #1 p1
  set var rown to .rown + 1
endwhile
write "_____" at .rown,57
set var rown to .rown + 2
write .total at .rown,10
write .ttotal at .rown,57
write .cont at 23,5
pause
label quit
remove consold
cls
clear tpaid tsold tmem answer fname rown ttotal tquant tuprice total
+header
return


$COMMAND
AUPD
*(MODULE SUMMARIZES TOTAL AMOUNT OWED FOR CONSIGMENTS WHICH HAVE
BEEN SOLD)
*(WRITTEN BY GARY RADKE)
*(FOR PEOPLES GROCERY DATABASE)
cls
set messages off
set error messages off
set var tpaid to n              *(conditional comparison variables)
set var tsold to y              *(                                 )
set var tmem text               *(member id variable)
```

```
set var fname to text           *(members full name)
set var rown to 8               *(row position variable)
set var ttotal currency                 *(total value of unpaid
consignments)
set var tquant integer              *(number of a given consignment
item)
set var tuprice currency         *(unit price of a given consignment
item)
set var header to "SUMMARY OF SOLD UNPAID CONSIGNMENTS"
set var total to "TOTAL SOLD BUT UNPAID CONSIGNMENTS"
set var ttotal = 0
set var cont to "press any key to continue"
label entnum
*(create a temporary file containing only records of sold but
unpaid)
*(consignment items)
project allupaid from member_c using all sorted by mem_no where
mempaid eq +.tpaid and all_sold eq .tsold
run achdr in consmod
set var tmem to mem_no in allupaid
set pointer #2 p2 for allupaid where mem_no eq .tmem
while p2 eq 0 then
*(find the name of the current member in question)
   set var tfirst to firstnam in member where mem_no eq .tmem
   set var tlast to lastname in member where mem_no eq .tmem
   set var fname to (.tfirst & .tlast)
   set pointer #1 p1 for allupaid where mem_no eq .tmem
   *(total all consignments sold but for which the member has not
  been paid)
   while p1 eq 0 then
     set var tquant to no_of_it in #1
     set var tuprice to consgrpr in #1
     set var tvalue = .tquant * .tuprice
     set var ttotal = .ttotal + .tvalue
     write .fname at .rown,2
     write .tvalue at .rown,47
     next #1 p1
     set var rown to .rown + 1
   endwhile
   set pointer #2 p2 for allupaid where mem_no gt .tmem
   set var tmem to mem_no in #2
endwhile
write "_____" at .rown,47
set var rown to .rown + 2
write .total at .rown,6
write .ttotal at .rown,47
label quit
remove allupaid
write .cont at 23,5
pause
cls
clear tpaid tsold tmem fname rown ttotal tquant tuprice cont header
```

```
return


$COMMAND
AUNPD
*(MODULE SUMMARIZES TOTAL AMOUNT OWED TO MEMBERS FOR CONSIGNMENTS
BOTH SOLD)
*(AND UNSOLD)
*(WRITTEN BY GARY RADKE)
*(FOR PEOPLES GROCERY DATABASE)

cls
set messages off
set error messages off
set var tpaid to n       *(conditional comparison variables)
set var tsold to y       *(                                 )
set var tmem text        *(member id # variable)
set var fname to text    *(variable for full member name)
set var rown to 8        *(row position designator)
set var ttotal currency  *(total value of consignments)
set var tquant integer   *(number of a given item on consignment)
set var tuprice currency *(cost of a consignment item)
set var header to "SUMMARY OF UNPAID CONSIGNMENTS SOLD AND UNSOLD"
set var total to "SUMMARY OF TOTAL UNPAID CONSIGNMENTS"
set var ttotal = 0
set var tmemval currency *(value of a members consignment items)
set var tmemval = 0
set var cont to "press any key to continue"
project allupaid from member_c using all sorted by mem_no where
mempaid eq +
.tpaid
run achdr in consmod
set var tmem to mem_no in allupaid
set pointer #2 p2 for allupaid where mem_no eq .tmem
while p2 eq 0 then
*(find the name of the current member under consideration)
   set var tfirst to firstnam in member where mem_no eq .tmem
   set var tlast to lastname in member where mem_no eq .tmem
   set var fname to (.tfirst & .tlast)
   set pointer #1 p1 for allupaid where mem_no eq .tmem
   *(total the value of consignments for this member)
   while p1 eq 0 then
     set var tquant to no_of_it in #1
     set var tuprice to consgrpr in #1
     set var tvalue = .tquant * .tuprice
     set var tmemval = .tmemval + tvalue
     set var ttotal = .ttotal + .tvalue
     next #1 p1
   endwhile
   write .fname at .rown,2
   write .tmemval at .rown,47
   set pointer #2 p2 for allupaid where mem_no gt .tmem
```

```
   set var tmem to mem_no in #2
   set var rown to .rown + 1
   set var tmemval = 0
endwhile
write "_____" at .rown,47
set var rown to .rown + 2
write .total at .rown,6
write .ttotal at .rown,47
remove allupaid
write .cont at 23,5
pause
cls
clear tpaid tsold tmem fname rown ttotal tquant tuprice total
tmemval cont +
header
return


$COMMAND
COMBORD
*(SUMMARIZES AN ORDER SUBMITTED TO A USER SELECTED SUPPLIER ON A
USER)
*(SELECTED DATE.  DISPLAYS PRODUCTS ORDERED, QUANTITY, AND VALUE)
*(WRITTEN BY GARY RADKE)
*(FOR PEOPLES GROCERY DATABASE)

cls
set messages off
set error messages off
set var tdescrip text
set var tquant real
set var tupr currency
set var totquant real
set var rown integer
set var itval currency
set var rown to 9
set var tquant = 0.0
set var totquant = 0.0
set var itval = 0.0
set var totval currency
set var totval = 0.0
set var cont to "press any key to continue"
label entrdate
*(get the date of the order to summarize)
run entdate in ordlabs
*(if no date is entered, report the error and allow the user to
reenter)
*(or exit the module)
if tdate fails then
  if answer ne x then
    goto entrdate
  else
```

```
    goto quit
  endif
endif
cls
write "PLEASE WAIT" AT 2,5
*(create a temporary file containing order dates and suppliers)
project orders from item_ord using date, supname sorted by date
delete duplicates from orders
*(if no order was submitted on the entered date, report the error
and)
*(let the user reenter)
set pointer #1 p1 for item_ord where date eq .tdate
if p1 ne 0 then
  run noord in ordlabs
  goto entrdate
endif

label entsup
set var answer to " "
*(allow the user to enter the name of the suppler for the order to
be)
*(summarized)
run entsup in ordlabs
*(if no supplier is entered, report the error and let the user
reenter)
*(or exit this module)
if vsupname fails then
  if answer ne x then
    goto entsup
  else
    goto quit
  endif
endif
set pointer #1 p1 for item_ord where date eq .tdate and supname eq
.vsupname
*(if no order was submitted to the given supplier on the given date,
report)
*(the error, display order dates and suppliers and let the user
reenter)
if p1 ne 0 then
  run nosup in ordlabs
  goto entsup
endif
remove orders
*(create a temporary file containing only those records in the
specified order)
project storeord from item_ord using all sorted by item_no where
date eq + .tdate and supname eq .vsupname
set var titem to item_no in storeord
*(find the first record of the temporary file)
set pointer #1 p1 for storeord where item_no eq .titem
cls
```

```
run combhdr in combo
while p1 eq 0 then
   set pointer #2 p2 for storeord where item_no eq .titem
   while p2 eq 0 then
     *(step through the temporary file and tally the total amount of
     an ordered item and its total value)
     set var tquant to quantity in #2
     set var totquant = .totquant + .tquant
     set var tupr to uprice in prodcat where item_no eq .titem
     set var itval = .itval + (.tquant * .tupr)
     next #2 p2
   endwhile
   *(write the item #, description, total quantity ordered, and
total value)
   set var totval = .totval + .itval
   write .titem at .rown,4
   set var tdescrip to descript in prodcat where item_no eq .titem
   set pointer #1 p1 for storeord where item_no gt .titem
   set var titem to item_no in #1
   write .tdescrip at .rown, 14
   write .tquant at .rown, 50
   write .itval at .rown, 61
   set var rown to .rown +1
   set var itval = 0.0
   set var totquant = 0.0
   set var tquant = 0.0
endwhile
*(report total value of order before equity and other added costs)
write "==========" at .rown,61
set var rown to .rown + 1
write "TOTAL VALUE OF ORDER" AT .rown,20
write .totval at .rown,61
label quit
write .cont at 23,1
pause
clear tdescrip tquant tupr totquant rown itval
remove storeord
return



$COMMAND
LISTEX

*(MODULE SUMMARIZES EXPENSES FOR ANY MONTH DESIGNATED BY THE USER)
*(EXPENSE DESCRIPTIONS AND AMOUNTS ARE LISTED AS WELL AS TOTAL OF )
*(GIVEN MONTHS EXPENSES)
cls
set messages off
set error messages off
set var tdate date          *(period for expense summarization)
```

```
set var tamt currency       *(expense amount variable)
set var tdescrip text       *(expense description)
set var rown integer        *(row location)
set var dcol integer        *(description column location)
set var acol integer        *(amount column location)
set var tot currency        *(total expense amount )
set var tamout currency
set var rown = 8
set var dcol = 7
set var dmess to "DESCRIPTION"
set var amess to "AMOUNT"
set var cont to "press any key to continue"
set var summary to "SUMMARY OF ALL EXPENSES FOR MONTH OF "
set var space to " "
set var acol = 55
s   e   t       v   a   r       l   i   n   e       t   o
"_____________________________________________________+
_____________"
set var sline to "_____________"
label entperio
write "ENTER MONTH AND YEAR OF PERIOD YOU WANT SUMMARIZED" AT 3,5
set var mess to ("mm/dd/yy   ")
fillin tdate using .mess at 3,56
if tdate fails then
   goto entperio
endif

cls
*(create a new file containing only expenses occuring in the month
and year of the date entered above)
project expe from expenc using date, amount, paid_to, descript
sorted by date + where imon(date) eq imon(.tdate) and iyr(date) eq
iyr(.tdate)
set var tamt = 0.0
set var tot = 0.0
set pointer #1 p1 for expe where amount gt .tamt
set var header to (.summary & tmon(.tdate))
set var tyear to iyr(.tdate)
write .line at 1,1
write .header AT 3,17
write .tyear at 3,65
write .line at 4,1
write .dmess at 5,.dcol
write .amess at 5,.acol
write .line at 6,1
*(step through the new file and print the description and amount of)
*(all of the records it contains. Keep a running total and print the
total)
*(at the end of the report)
while p1 eq 0 then
 set var tdescrip to descript in #1
 set var tamout to amount in #1
```

```
 write .tdescrip at .rown, .dcol
 write .tamout at .rown, .acol
 set var tot to .tot + .tamout
 next #1 pl
 set var rown to .rown + 1
endwhile
set var rown to .rown + 1
write "_____" at .rown,.acol
set var rown to .rown + 2
write "TOTAL EXPENSES FOR THE MONTH" AT .rown,12
write .tot at .rown,.acol
write .cont at 23,5
pause
remove expe
clear tdate tamt tdescrip town dcol acol tot tamout dmess amess cont
summary +
space line sline
return



$COMMAND
OD
*(MODULE SUMMARIZES TOTAL ORDERS SUBMITTED TO SUPPLIERS AND ALL
EXPENSES)
*(FOR ANY GIVEN MONTH IN A YEAR SPECIFIED BY THE USER)


cls
set messages off
set error messages off
set var line text
set var header text
set var titem integer       *( item number variable)
set var tdate date          *( date variable)
set var tsales currency     *( total sales variable)
set var texp currency       *( total expenses)
set var cost currency       *( product cost )
set var tsup to text        *( supplier name variable)
set var tamount real        *( number of items purchased)
set var tsales to 0.0
set var cost to 0.00
set var tamount to 0
set var cexp currency       *( temporary expense variable)
set var cexp to 0.00
set var texp to 0.00
set var header to "SALES / EXPENSE REPORT FOR THE MONTH OF "
set var sal to "TOTAL VALUE OF GOODS ORDERED FROM SUPPLIERS THIS
MONTH "
set var exhdr to "TOTAL MONTHLY EXPENSES"
s   e   t       v   a   r       l   i   n   e       t   o
"_____+
```

```
_____"
set var cont to "press any key to continue"
*( get the date of the period the user wants summarized)
label entdate
write "Enter date to summarize"  at 3,5
fillin tdate using "mm/dd/yy  "  at 4,5
if tdate fails then
   goto entdate
endif
set var header to (header & tmon(.tdate))
set var tyear to iyr(.tdate)
cls
write "PLEASE WAIT" AT 3,5
*(create a temporary file containing only sales records within the
period specified above)
project sales from item_ord using all sorted by item_no +
where imon(date) eq imon(.tdate) and iyr(date) eq iyr(.tdate)
set var tamount to quantity in sales
set pointer #1 pl for sales where quantity  gt 0
*(step through new file and total period sales)
while pl eq 0 then
  set var tamount to quantity in #1
  set var tsup to supname in #1
  set var titem to item_no in #1
  set var cost to uprice in prodcat where item_no eq .titem
  set var ocost = .cost * .tamount
  set var tsales = .tsales + .ocost
  next #1 pl
endwhile
*(create a temporary file containing only expense records within the
period specified above)
project aexp from expenc using all where imon(date) eq imon(.tdate)
+ and iyr(date) eq iyr(.tdate)
set var cexp to amount in aexp
set pointer #1 pl for aexp where amount gt 0
*(step through new expense file and total all expenses)
while pl eq 0 then
  set var cexp to amount in #1
  set var texp = .texp + .cexp
  next #1 pl
endwhile
cls
write .cont
pause
cls
write .line at 2,1
write .header at 3,14
write .tyear at 3,64
write .line at 4,1
write .line at 5,1
write .sal at 7,5
write .tsales at 7,65
```

```
write .exhdr at 11,5
write .texp at 11,65
write .line at 15,1
remove sales
remove aexp
write .cont at 23,5
pause
clear titem tdate tsales texp cost tsup tamount cexp header sal
exhdr line + cont
return



$COMMAND
EXCON
*(PRINTS SUMMARY OF OUTSTANDING DEBTS INCLUDING SOLD AND UNSOLD
CONSIGNMENTS)


cls
set messages off
set error messages off
set var lcol to 5          *(left column location)
set var rcol to 50         *(right column location)
set var notp to "n"        *(not paid value for conditional)
set var pt text            *(paid to temporary variable)
set var des text           *(description temporary variable)
set var tamt currency      *(total amount owed)
set var rown integer       *(row designator)
set var rown to 9
set var tupexp currency    *(total unpaid expenses)
set var tupexp to 0.00
set var conpr currency     *(price consigner wants for product)
set var conpr to 0.00
set var tquant integer     *(number of a given item on consignment)
set var tval currency        *(total value of consignments a given
member has)
set var tval to 0.00
s   e   t       v   a   r       l   i   n   e       t   o
"_____________________________________________________+
_____________"
set var shrtlin to "_____________"
set var contmes to "press any key to continue"
set var hdr to "REPORT OF ALL OUTSTANDING DEBTS"  *(              )
set var exphdr to "UNPAID EXPENSES"                        *(   report
messages  )
set var conhdr to "UNPAID CONSIGNMENTS"           *(              )
set var ttot to "TOTAL "                          *(              )
set var cont to "press any key to continue"
write .hdr at 2,24
write .line at 3,1
write .exphdr at 5,5
```

```
write .line at 6,1
write .line at 7,1
*(create the file upexp containing only unpaid expenses)
project upexp from expenc using all where paid eq .notp
set pointer #1 p1 for upexp where amount gt 0
*(step through the new file printing the expense description, amount
and keep a running total of unpaid expenses)
while p1 eq 0 then
   set var pt to paid_to in #1
   set var des to descript in #1
   set var tamt to amount in #1
   set var tupexp = .tupexp + .tamt
   write .des at .rown,.1col
   write .tamt at .rown,.rcol
   next #1 p1
   set var rown to .rown + 1
endwhile
set var mess to (.ttot & .exphdr)
write .shrtlin at .rown,47
set var rown to .rown + 1
write .mess at .rown,15
write .tupexp at .rown,.rcol
set var .rown to 9
write .cont at 23,3
pause
cls

write .line at 3,1
write .conhdr at 5,5
write .line at 6,1
write .line at 7,1
*(create a new file containing only consignments for which members
have not paid.)
project upconsg from member_c using all sorted by mem_no where
mempaid eq + .notp
set var tmem to mem_no in upconsg
set pointer #1 p1 for upconsg where mem_no eq .tmem
set var tamt to 0.00
*(step through the file totaling the unpaid consignment values for
each member and print them)
while p1 eq 0 then
   set var tval to 0.00
   set var fnam to firstnam in member where mem_no eq .tmem
   set var lnam to lastname in member where mem_no eq .tmem
   set var fulnam to (.fnam & .lnam)
   set pointer #2 p2 for upconsg where mem_no eq .tmem
   while p2 eq 0 then
      set var conpr to consgrpr in #2
      set var tquant to no_of_it in #2
      set var vtcon = .conpr * .tquant
      set var tval = .tval + .vtcon
      next #2 p2
```

```
   endwhile
   set var rown to .rown + 1
   set var tamt to .tamt + .tval
   write .fulnam at .rown .lcol
   write .tval at .rown, .rcol
   set pointer #1 pl for upconsg where mem_no gt .tmem
   set var tmem to mem_no in #1
endwhile
set var rown to .rown + 1
*(display total unpaid consignment value)
write .shrtlin at .rown,47
set var rown to .rown +1
set var mess to (.ttot & .conhdr)
write .mess at .rown, 15
write .tamt at .rown, .rcol
*(remove temporary tables)
remove upexp
remove upconsg
write .cont at 23,5
pause
set messages on
set error messages on
return



$COMMAND
PRTLBL
*(PROGRAM PRINT LABELS)
*(PRINTS PRODUCT BREAKDOWN LABELS FOR A GIVEN ORDER)
cls
set messages off
set error messages off

SET VAR rown TO 8
SET VAR tdate date
SET VAR vsupname TO text
set var tdescrip text
set var err text
set var answer to " "
LABEL entrdate
*(find date of order for which to print labels)

RUN entdate IN ordlabs
*( if no date is entered let user retry or quit module)
if tdate fails then
   if answer ne x then
      goto entrdate
   else
      goto quit
   endif
```

```
endif
*(make a temporary file of all order dates and suppliers)

project orders from item_ord using date, supname sorted by date
delete duplicates from orders
*(if entered order date doesn't exist in temporary file, display
error message, display all order dates and suppliers and let user
reenter)
set pointer #1 p1 for orders where date eq .tdate
if p1 ne 0 then
   run noord in ordlabs
   goto entrdate
endif

CLS
set var answer to " "
label entrsup
*(find supplier for which to print order distribution labels)
run entsup in ordlabs
*(if no supplier is entered display error message and let user
reenter)

if vsupname fails then
   if answer ne x then
      goto entrsup
   else
      goto quit
   endif
endif
set pointer #1 p1 for orders where date eq .tdate and supname eq
.vsupname
*(if no order was placed to entered supplier on entered date,
display error)
*(message, display order dates and suppliers, and let user reenter)

if p1 ne 0 then
   run  nosup in ordlabs
   goto entrsup
endif
*(delete temporary file)
remove orders
*(create a temporary file containing only those records matching the
entered)
*(order date and supplier)

PROJECT ordlab FROM item_ord USING ALL SORTED BY item_no WHERE date
EQ + .tdate AND supname EQ .vsupname
CLS
SET VAR nit TO item_no IN ordlab
SET POINTER #1 p1 FOR ordlab WHERE item_no EQ .nit
*(step through the temporary file by item number and print member
names and the amount they ordered)
```

```
WHILE p1 EQ 0 THEN
  SET VAR vit to item_no IN #1
  SET POINTER #3 p3 FOR ordlab WHERE item_no EQ .nit
  set var tdescrip to descript in prodcat where item_no eq .nit
  *(print the order label header)
  run lblhdr in ordlabs
  WHILE p3 EQ 0 THEN
    SET VAR vmem TO mem_no in #3
    set pointer #2 p2 for member where mem_no eq .vmem
    set var vfirst to firstnam in #2
    set var vlast to lastname in #2
    set var fname to (.vfirst & .vlast)
    set var vquant to quantity in #3
    write .fname at .rown, 10
    write .vquant at .rown,50
    set var rown to .rown +1
    next #3 p3
  endwhile
  write "PRESS ANY KEY TO CONTINUE " AT 23,1
  pause
  cls
  *(find the next item number in the temporary file)
  set pointer #1 p1 for ordlab where item_no gt .vit
  set var nit to item_no in #1
  set var rown to 8
endwhile
label quit
Remove ordlab
clear rown tdate vsupname tdescrip err answer
return


$COMMAND
PB
*(MODULE PRINTS AN ITEMIZED BILL FOR EACH MEMBER PARTICIPATING IN A)
*(GIVEN ORDER.  REPORTS ORDER DATE, SUPPLER CATALOG #, DESCRIPTION)
*(QUANTITY AND PRICE FOR EACH ITEM ORDERED.  MEMBER EQUITY, SHIPPING
COST, MARKUP AND TAX ARE ADDED TO ORDER TOTAL.)
cls
set messages off
set error messages off
set var answer text
set var ctax real              *(tax rate variables)
set var ttax real
set var cmarkup real           *(markup variables)
set var tmarkup real
set var cequity real           *(member equity variables)
set var tequity real
set var cship currency         *(shipping cost variables)
set var tship currency
set var tdate date             *(order date)
set var nmem to text           *(member name)
```

```
set var vsupname to text      *(supplier name)
set var tfirst to text         *(first name of member)
set var tlast to text          *(last name of member)
set var fname to text          *(full name of member)
set var tprice currency        *(total cost of goods)
set var tdescrip to text       *(description of item)
set var vmem to text           *(member #)
set var rown to 8              *(row designator)
set var tcat integer           *(item catalog number)
set var tupr currency          *(unit price of item)
set var tcas currency          *(case price of item)
set var tswt integer           *(shipping weight of item)
set var upcas integer          *(units per case of an item)
set var uwt real               *(unit weight)
set var itwt integer           *(weight per item)
set var tval currency
set var totwt integer          *(total weight of order)
set var vmarkup currency       *(amount of markup)
set var tquant real            *(quantity of an item ordered)
set var totwt = 0
set var cont to "press any key to continue"
set var taxm to "SALES TAX "
set var markup to "MARKUP "
set var bpe to "MEMBER EQUITY "
set var shipping to "SHIPPING COST PER POUND "
set var ctax = 4.5
set var cmarkup = 15
set var cequity = 2
set var cship = .02
label enterdate
*(get date of order)
run entdate in pbl
*(if no date is entered, report error and let user reenter or exit
module)
if tdate fails then
  if answer ne x then
    goto enterdate
  else
    goto quit
  endif
endif
cls
write "PLEASE WAIT" AT 2,5
*(create a temporary file containing both order and catalog
information)
intersect prodcat with item_ord forming prodord using cas_pric
item_no +
shipwt uprice supname mem_no date quantity descript
*(create a temporary file containing order dates and suppliers)
project orders from item_ord using date, supname sorted by date
delete duplicates from orders
```

```
*(check to see if an order was submitted on the day entered. If not,
display order dates and suppliers and report the error allow
reentry)
set pointer #1 p1 for item_ord where date eq .tdate
if p1 ne 0 then
  run noord in pbl
  goto enterdate
endif
*(enter name of order supplier.  )
label entrsup
run entsup in pbl
*(if no supplier is entered report the error and give the user an
opportunity)
*(to cancel the operation otherwise give them another chance)
if vsupname fails then
  if answer ne x then
    goto entrsup
  else
    goto quit
  endif
endif
set pointer #1  p1 for item_ord where date eq .tdate and supname eq
.vsupname
*(if no order was submitted to this suppler on this date report the
error)
*(display order dates and suppliers and let the user reenter)
if p1 ne 0 then
  run nosup in pbl
  goto entrsup
endif
remove orders
*(display current tax rate, get update if necessary)
run gtax in pbl
*(display current markup rate, get update if necessary)
run gmarkup in pbl
*(display current equity charge, get update if necessary)
run gequity in pbl
*(display current shipping cost rate, get update if necessary)
run gship in pbl
cls
*(create a temporary file containing only those records involving
the)
*(date and supplier entered above)
project prbil from prodord using all sorted by mem_no item_no where
date eq + .tdate and supname eq .vsupname
set var answer to "a"
run adjust in pbl
set var nmem to mem_no in prbil
if answer eq x then
   goto quit
endif
*(compile the full member name )
```

```
set pointer #1 p1 for prbil where mem_no eq .nmem
set var tequity = .cequity / 100
set var tmarkup = .cmarkup / 100
set var ttax = .ctax / 100
while p1 eq 0 then
   set var tfirst to firstnam in member where mem_no eq .nmem
   set var tlast to lastname in member where mem_no eq .nmem
   set var fname to (.tfirst & .tlast)
   set var tval = 0
   run bilhdr in pb1
   set var vmem to mem_no in #1
   set pointer #2 p2 for prbil where mem_no eq .vmem
   *(itemize order for member)
   while p2 eq 0 then
     set var tcas to cas_pric in #2
     set var tupr to uprice in #2
     set var tswt to shipwt in #2
     set var upcas = .tcas / .tupr
     set var uwt = .tswt / .upcas
     set var tcat to item_no in #2
     set var tdescrip to descript in #2
     set var tquant to quantity in #2
     set var tupr to uprice in #2
     set var itwt = .uwt * tquant
     set var totwt = .totwt + .itwt
     set var tprice = .tupr * .tquant
     set var tval = .tval + .tprice
     write .tcat at .rown,2
     write .tdescrip at .rown,15
     write .tquant at .rown,50
     write .tprice at .rown,65
     set var rown to .rown + 1
     next #2 p2
   endwhile
   write "_____________" at .rown,65
   set var rown to .rown + 1
   set var mess1 to "VALUE OF ORDER "
   write .mess1 at .rown,22
   write .tval at .rown,65
   set var rown to .rown + 1
   set var shipcost = .totwt * .cship
   set var equicost = .tval * .tequity
   set var mess2 to "MEMBER EQUITY CHARGE "
   write .mess2 at .rown,22
   write .equicost at .rown,65
   set var rown to .rown + 1
   set var vmarkup = .tmarkup * .tval
   set var m3 to "MARKUP (PRE EQUITY TOTAL X  " & ctxt(.cmarkup) &
"percent)"
   write .m3 at .rown,22
   write .vmarkup at .rown,65
   set var rown to .rown + 1
```

```
    set var tval = .tval + .vmarkup
    set var txval = .tval * .ttax
    set var mess4 to "TAX " & ctxt(.ctax) & "percent"
    write .mess4 a .rown,22
    write .txval at .rown,65
    set var rown to .rown +1
    set var mess5 to "SHIPPING COST"
    write .mess5 at .rown,22
    write .shipcost at .rown,65
    set var rown to .rown + 1
    set var tval = .tval + .txval
    set var tval = .tval + .equicost
    set var tval = .tval + .shipcost
    set var mess6 to "TOTAL CHARGE"
    write "_____" at .rown,65
    set var rown to .rown + 1
    write .mess6 at .rown,30
    write .tval at .rown,65
    set pointer #1 p1 for prbil where mem_no gt .vmem
    set var nmem to mem_no in #1
    set var rown to 8
    write .cont at 23,5
    pause
    set var totwt = 0
    set var tval =0.0
    set var vmarkup = 0.0
    set var shipcost = 0.0
    set var equicost = 0.0
    cls
endwhile
label quit
remove prbil
remove prodord
clear ctax ttax cmarkup tmarkup cequity tequity cship tship tdate
nmem vsupname tfirst tlast fname tprice tdescrip vmem rown tcat tupr
tcas tswt upcas uwt itwt tval totwt vmarkup tquant
cls
return
```

APPENDIX C:DATA DICTIONARY

ADDRESS: The mailing address of a member or supplier.

ALL_SOLD: A boolean that indicates if all units of a given item placed on consignment have been sold.

AMOUNT: Dollar amount paid for an expense.

CASPRICE: Price of the minimum quantity of an item that can be ordered from a supplier.

CONSGRPRICE: The price a member wants to receive for an item he/she has placed on consignment.

CONTACT: The name of a suppliers representative with whom the cooperative deals.

DATE: Month/day/year of a transaction, be it an order, an expense payment, or placing an item on consignment.

DESCRIPT: Description of a product being ordered or placed on consignment.

DESCRIPTIO: Description of goods or services the cooperative receives which are not resold. i.e. insurance, utilities, equipment repairs.

FIRSTNAME: A members first name.

HOMEPHONE: A members home phone number.

ITEMNO: A catalog number associated with products offered by various suppliers. Numbers are unique within the products offered by any single supplier but may be duplicated by different suppliers.

LASTNAME: A members last name.

MEM_NO: A unique number assigned to each member that joins the cooperative.

NO_OF_ITEM: The number of items of a given description a member has placed on consignment on a given date.

PAID: A boolean that when true designates that an expense has been paid.

PAIDTO: The name of an individual, or organization receiving money from the cooperative for services or goods not destined for resale. i.e. advertising, mailing costs, utilities.

PHONE: A suppliers business phone number.

QUANTITY: The number of units of an item a member has ordered.

SHIPWT: The shipping weight of the smallest unit a supplier will sell.

STATE: The abbreviation for the state in which a member resides or a supplier is located

SUPNAME: The names of companies or individuals that supply goods to the cooperative.

TOWN: The town in which a supplier is located or a member resides.

UNITPRICE: Price / unit charged by a supplier, where a unit may be a pound, can, jar, packet, etc.

WORKPHONE: A members phone number at work, if any.

ZIPCODE: The five digit postal service zipcode for a members or suppliers address.

A COMPARISON OF TWO MICRO COMPUTER DATABASE MANAGEMENT SYSTEM PRODUCTS

by

Gary A. Radke

B.S., Kansas State University, 1979

AN ABSTRACT OF A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

The software market is currently crowded with micro computer based database management systems. The complexity of the systems available ranges from simple information management systems which are inflexible in their format to the advanced systems with programmable products that allow the user to easily write an application tailored to their precise needs. The simple systems force the user to adapt their application to the structure of the database management system; the advanced systems often do not require the user to learn a great deal more about database management systems.

This report will discuss the creation of an extensible, "user friendly" front end for the database of a small enterprise. A user interface has been developed using both dBASE III+ and RBASE 5000 with the object of providing qualitative information on the effort required to establish the database and produce a user interface for each of these systems.

The need for practical information when evaluating a database management system, the enterprise to be implemented, the operation of the applications developed in dBASE III+ and RBASE 5000 are discussed. In addition, the development of the two applications and various aspects of the process are presented. Areas for possible future study are also suggested.