Spring 5-22-2017

A LTIHub for Composite Assignments

Sunita Rajain
San Jose State University

Recommended Citation

Rajain, Sunita, "A LTIHub for Composite Assignments" (2017). *Master's Projects*. 525.
DOI: https://doi.org/10.31979/etd.aa7y-hsvh
https://scholarworks.sjsu.edu/etd_projects/525

A LTIHub for Composite Assignments

A

Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

By Sunita Rajain

May 2017

The Designated Project Committee Approves the Project Titled

A LTIHub for Composite Assignments

By

Sunita Rajain

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Cay Horstmann, Department of Computer Science Date

Professor Thomas Austin, Department of Computer Science Date

Professor Ronald Mak, Department of Computer Engineering Date

ACKNOWLEDGEMENTS

It would have been very difficult for me to complete my project without the help and motivation from various dignitaries. I have an immeasurable pleasure to present my acknowledgement and gratitude to everyone who has given me his or her hand in this project.

I would like to take this opportunity to thank my project advisor, Dr. Cay Horstmann, for his continuous guidance and encouragement and with gratitude to Dr. Thomas Austin and Professor Ron Mak, for devoting their valuable time and being my committee members.

I feel honored that both Professor Ronald Mak and Professor Kong Li put an extra effort in giving their students practice exercises using LTIHub and also granted me permission to conduct survey and collect data to measure effectiveness of LTIHub in student's learning.

ABSTRACT

Learning management systems (LMS) such as Canvas and Blackboard use Learning Tool Interoperability (LTI) as their main integration point for external learning tools. Each external tool provider has to implement LTI specifications or follow LTI standards that is a time consuming and complex process as there is no easy to follow specification available. Through this project, I have developed a system that follows the LTI specifications and integrates the CodeCheck autograder and interactive exercises with any LMS. I developed a Java based web app named LTIHub that acts as a mediator between LMS and any Learning Tool Provider. The LTIHub takes care of the LTI specification. The Learning Tool Provider needs to provide URL for each problem and a callback URL. This application stores the information related to the status of assignments and passes the information back to the LMS when all problems of an assignment are done by a student. I have prepared this report on the architecture of the software and analysis of students data during a semester of deployment.

TABLE OF CONTENTS

List of Figures

List of Acronyms and Terms

LMS - Learning Management System

LTI - Learning Tool Interoperability

TP - Tool Provider

TC - Tool Consumer

URL - Uniform Resource Locator

HTML - Hyper Text Markup Language

JSON - JavaScript Object Notation

XML - eXtensible Markup Language

AWS - Amazon Web Services

EC2 - Elastic Compute Cloud

GCE - Google Compute Engine

AMI - Amazon Machine Image

PPA - Personal Package Archives

MVC - Model View Controller

ELB - Elastic Load Balancing

1. Introduction

There are numerous studies and research that point out the issues faced by students in introductory programming courses and the reasons why so many students drop out of CS courses. A major problem faced by the instructors with their students in programming courses is that most students are not able to write simple code, such as for loop, switch statement etc. The students have the theoretical knowledge of concepts but in writing actual code, they fail to implement that. Students in introductory programming courses are novice programmers. They fail to see the “big picture”. Zingaro [2005] analyzes exam in CS1 courses that includes four theoretical problems with one coding question. The coding question requires knowledge as well as correct application of a number of concepts. Rubric for such question is difficult to create and use. So, evaluation of this question doesn’t provide accurate, informative feedback to students as it’s difficult to tell whether the student failed due to lack of conceptual knowledge or student was not able to design. Sequences of small questions that emphasize individual concepts can be a more effective feedback tool.

Vihavainen [2014] suggests that programming is not a single skill but a combination of many complex activities. He has analyzed various methods of teaching and concluded that cooperative learning and visual programming tools improve the students pass percentage in introductory programming courses. Relatable contents learning improve the students’ performance in introductory programming courses. Lahtinen [2005] paper survey analysis suggests that the large classes and differences in student backgrounds make it difficult to design the programming instruction that is beneficial for everyone.The concepts that require understanding of program’s large entities are difficult to understand by students. The novice programmers understand the basic concepts but don’t know how to apply them. If more practical learning materials like short coding problems, code tracing or code rearranging exercises are provided, then the students are expected to learn more and perform better.

Simon [2013] argues whether the programming problems used in 1980s can be used to evaluate students in present time by using the Soloway’s rainfall problem which is a variation of averaging a collection of numbers. Simon replicated previous studies in a computer-based setting and found that majority of students are still making the same type of errors as made by students in 1980s. The problem is not because students don’t have the conceptual knowledge but lack of practice that includes learning topic-by-topic to build a strong background for a more complex program that requires knowledge of many interrelated concepts.

Thomas [2001] suggests that a student’s perception of difficulty of task is different from the instructors as the student is unable to consider all the scenarios/situations. The more time it takes for a student to solve a practical work the more likely they will stop before completing the work. This suggests that it makes more sense if students are provided with couple of short programming problems in place of a long program in introductory CS courses. As per Ahadi [2016] paper there is a strong correlation between student’s score and how they get rid of an error in programming problem. Amount of time that a student spends on a given problem is strongly correlated to the final score of the student.

MOOC (Massive Open Online Course) is a model where online learning contents are made available to anyone who is interested in learning and doesn’t put any restriction on attendance. It is an emerging trend in technical education that provides a very effective learning platform. Zhenghao [2015] has analyzed the Coursera’s data of learners who have completed their courses and found that 72% respondents reported career benefits and 61% reported educational benefits. MOOC courses consist of recorded video tutorials, multiple choice questions, coding assignments etc. As per Pieterse [2013], MOOC assignments can be more useful or can have more positive impact on students' learning if fast and accurate feedback is also provided. This paper also summarizes some of the features that every MOOC should have like secure environment for student's program execution, automatic assessment, provision of resubmission, provide qualitative feedback, student's statistical data about the submission.

Sheard [2011] paper analyzes how the student's attitude towards plagiarism or cheating has changed over time by comparing responses of students to survey questionnaires in 2000 and 2010. Per this analysis, students are less inclined towards plagiarism in 2010 as compared to 2000. Though the inclination towards plagiarism is less, students in introductory computer classes are still cheating when they have the fear of failure and, workload and time pressure. As time progresses, the instructors as well as universities are becoming more intolerant about cheating. Edgcomb [2017] paper also states that students in introductory classes cheat when they are on the borderline of failure or they don't know what the question is asking them to do. Edgcomb has analyzed student behaviour in introductory programming classes to find how honestly the students are doing homework exercises. He concluded that the key factors to have honest and earnest completion of assignments are points for the assignments, properly designed course contents, and reasonable amount of work given to students.

The paper by Norris [2008] discusses a ClockIt toolset that monitors and logs student development activities while they work on programming problems so that the instructors can determine which activities help students to become successful developers. There is a wide difference in perception of programming practices between the instructors and students. As per the instructors, students don’t start early on their assignments, students consider compiling code to be a completely working code, student don’t test their code frequently, they write very long code before they start to compile, students don’t write robust and modular code. This tool helps to analyze student’s development activities so that the instructors can figure out who are struggling because of bad development practices. Instructor can use the analysis result of this tool to guide students to improve their performance and keep them interested in Computer Science and thereby reduce the dropout rate.

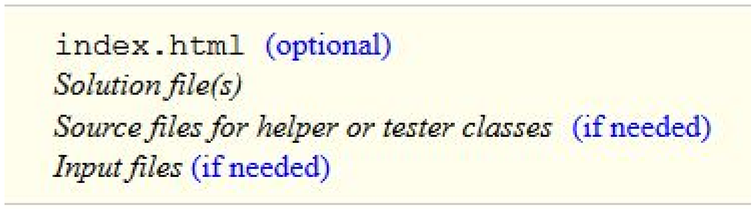
A majority of research papers and analysis results mentioned above suggests that students in introductory CS courses can have better learning if more short practice problems are provided to them. But due to the limited lecture sessions and a long list of topics to cover, it is not possible for the instructors to provide students with lots of in-class practice assignment for each and every topic. One solution is to recommend online programming practice tools to students because students learn more by actual coding than just reading or watching the lectures. Learning to code is just like learning mathematics, the more you practice, the better you get.The only issue with this approach is that the instructor has no idea which students are doing practice and what is each student’s progress.

To overcome this issue, a system is needed that can be integrated with the school’s existing LMS systems. This overcomes the headache of setting another system for the instructors and students to create account on another portal/application. The instructors can assign programming problems and keep track of student’s progress in a system they already run and use. Right now there is no such system available that allows instructors to create an assignment comprised of a combination of multiple coding problems and interactive exercises, keep track of students’ submissions and pass grades back to the LMS. The system should allow students to take breaks and later continue with the remaining problems of a particular assignment. I have developed a system that can be integrated with existing LMS systems and fulfills the other requirements as well.

2. CodeCheck And Interactive Exercises

2.1 CodeCheck

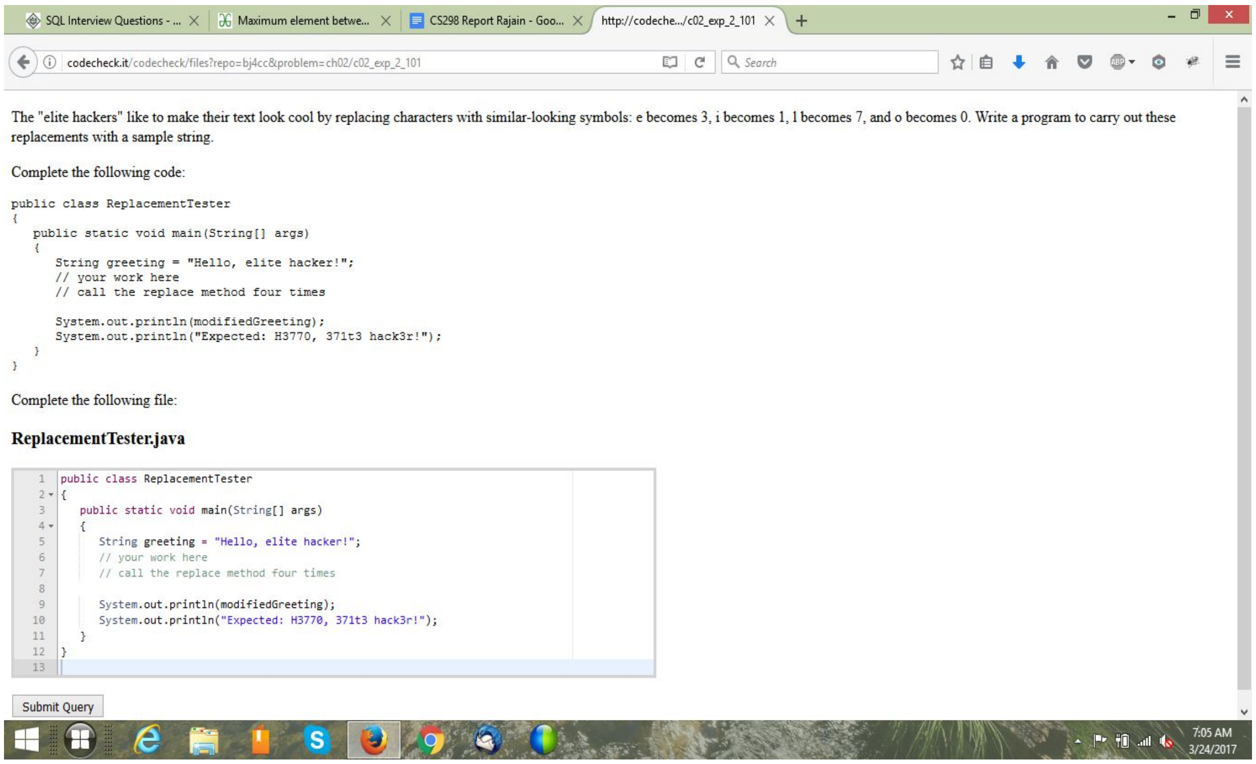
CodeCheck, developed by Cay Horstmann and his students, is a unit testing library that automatically evaluates solutions to programming problems submitted by students. Each CodeCheck problem is identified by a problem URL that is created by codecheck. Instructors can assign multiple CodeCheck problems to students by giving them problem URLs. The instructor uses the CodeCheck web interface to upload a problem zip file containing the following files for each problem:-



`index.html` (optional)
Solution file(s)
Source files for helper or tester classes (if needed)
Input files (if needed)

Figure 1 - Problem Zip Folder Contents to Upload on CodeCheck

A unique URL is generated after the files are uploaded on CodeCheck server. Students write a solution in the textbox provided by CodeCheck and submit the code. After checking the student submission, the CodeCheck server provides a score and evaluation report of the answer. Figure 2 and Figure 3 below shows a CodeCheck problem and a CodeCheck report.


SQL Interview Questions - ...
Maximum element betwe...
CS298 Report Rajain - Goo...
http://codeche.../c02_exp_2_101
codecheck.it/codecheck/files?repo=bj4cc&problem=ch02/c02_exp_2_101
Search
Submit Query
7:05 AM
3/24/2017

The "elite hackers" like to make their text look cool by replacing characters with similar-looking symbols: e becomes 3, i becomes 1, l becomes 7, and o becomes 0. Write a program to carry out these replacements with a sample string.

Complete the following code:

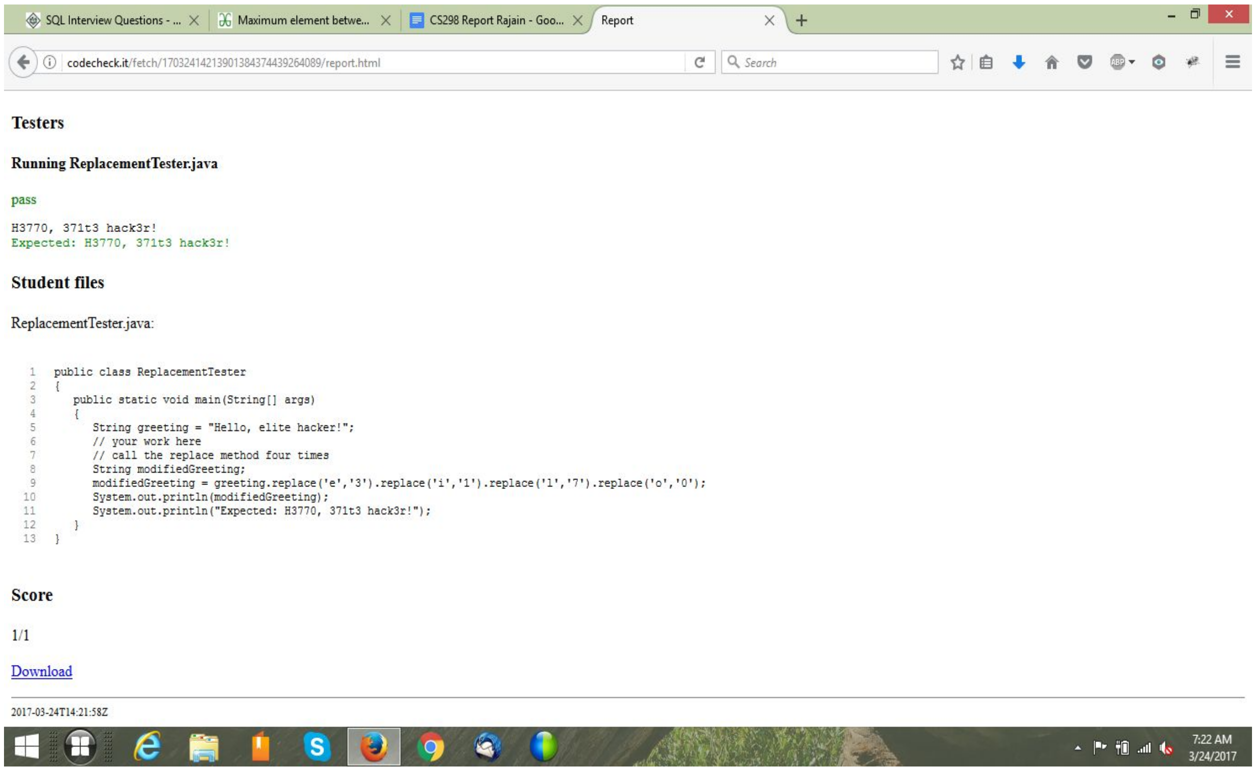
```
public class ReplacementTester
{
   public static void main(String[] args)
   {
      String greeting = "Hello, elite hacker!";
      // your work here
      // call the replace method four times

      System.out.println(modifiedGreeting);
      System.out.println("Expected: H3770, 371t3 hack3r!");
   }
}
```

Complete the following file:

ReplacementTester.java

```
public class ReplacementTester
{
   public static void main(String[] args)
   {
      String greeting = "Hello, elite hacker!";
      // your work here
      // call the replace method four times

      System.out.println(modifiedGreeting);
      System.out.println("Expected: H3770, 371t3 hack3r!");
   }
}
```

Figure 2 - CodeCheck Problem


SQL Interview Questions - ...
Maximum element betwe...
CS298 Report Rajain - Goo...
Report
codecheck.it/fetch/17032414213901384374439264089/report.html
Search
7:22 AM
3/24/2017

Testers

Running ReplacementTester.java

pass

```
H3770, 371t3 hack3r!
Expected: H3770, 371t3 hack3r!
```

Student files

ReplacementTester.java:

```java
public class ReplacementTester
{
   public static void main(String[] args)
   {
      String greeting = "Hello, elite hacker!";
      // your work here
      // call the replace method four times
      String modifiedGreeting;
      modifiedGreeting = greeting.replace('e','3').replace('i','1').replace('l','7').replace('o','0');
      System.out.println(modifiedGreeting);
      System.out.println("Expected: H3770, 371t3 hack3r!");
   }
}
```

Score

1/1

Download

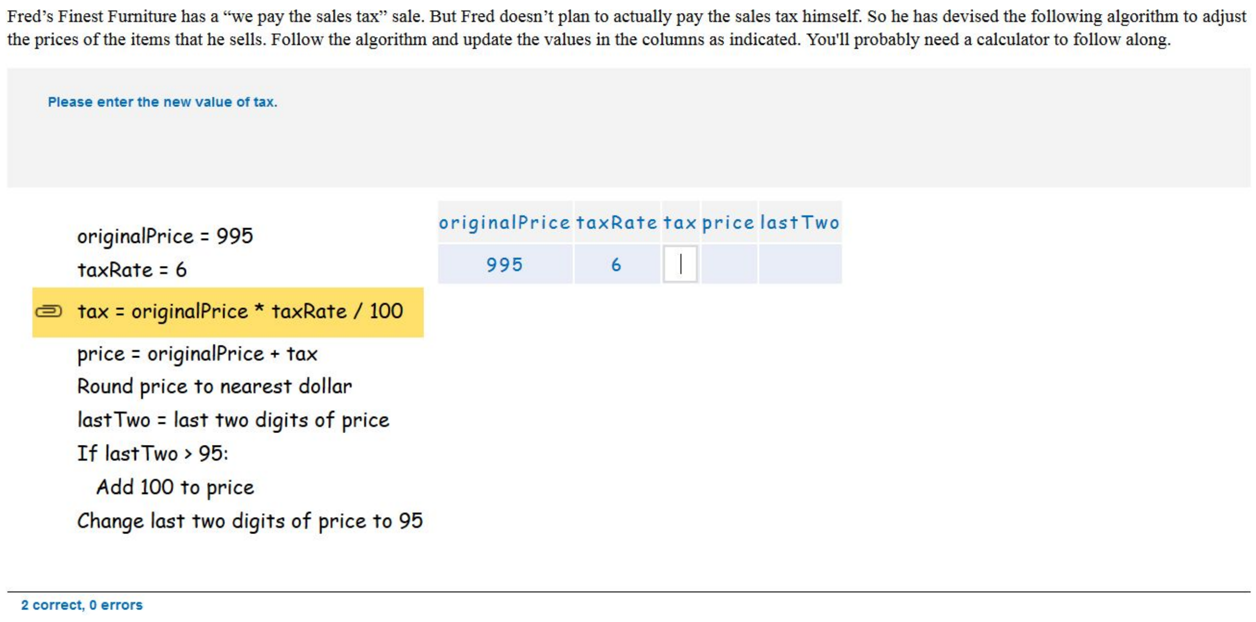
2017-03-24T14:21:58Z

Figure 3 - CodeCheck Report

2.2 Interactive Exercises

Cay Horstmann has designed interactive exercises for his electronic book- "*Big Java Late Objects" Interactive eBook*. Students study computer science concepts by reading a textbook or listening to lectures on Youtube, Edx, Coursera or other available platforms. They can also watch videos showing algorithm flow or graph or binary tree construction. However, watching videos is not as effective as providing a platform that lets students do the coding problem and get the evaluation at the same time. Interactive exercises help students to solidify their programming concepts by working on problems such as tracing code to understand the workflow, constructing linked list, or drawing trees and graphs.

Interactive exercises allow students to practice coding activities and provide immediate feedback and score. The Horstmann textbook uses four types of interactive exercises such as code tracing, object diagramming, knowledge-based exercises and rearranging code. Code tracing exercise requires students to enter value of variables and trace the flow of a program as shown in Figure 4.


Fred’s Finest Furniture has a “we pay the sales tax” sale. But Fred doesn’t plan to actually pay the sales tax himself. So he has devised the following algorithm to adjust the prices of the items that he sells. Follow the algorithm and update the values in the columns as indicated. You'll probably need a calculator to follow along.

Please enter the new value of tax.

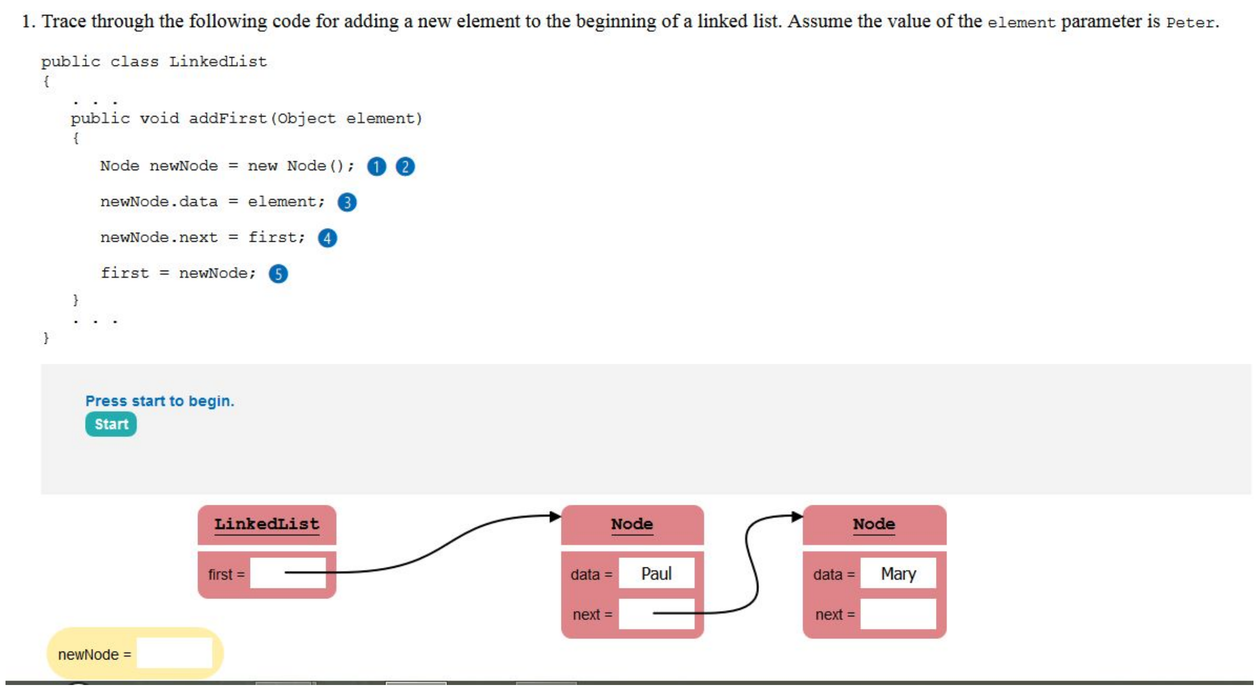
```
originalPrice = 995
taxRate = 6
tax = originalPrice * taxRate / 100
price = originalPrice + tax
Round price to nearest dollar
lastTwo = last two digits of price
If lastTwo > 95:
  Add 100 to price
Change last two digits of price to 95
```

originalPrice	taxRate	tax	price	lastTwo
995	6			

2 correct, 0 errors

Figure 4 - Code Tracing Exercise

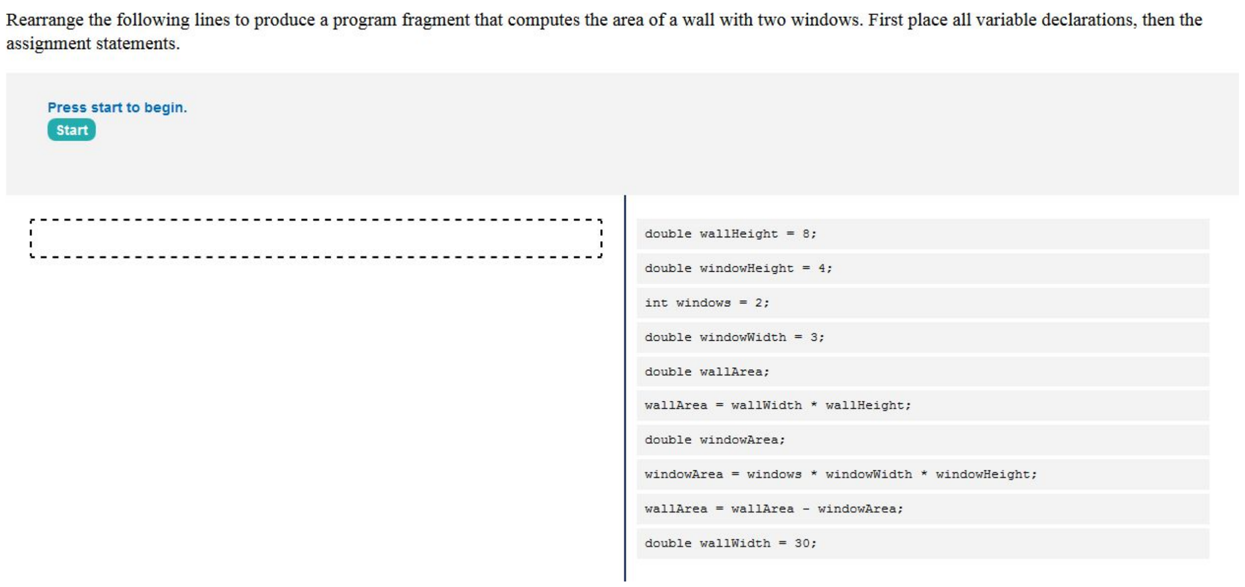
Object diagram exercises allow students to track stack variables and dynamically allocated objects. Figure 5 shows code and the related object diagram.


1. Trace through the following code for adding a new element to the beginning of a linked list. Assume the value of the `element` parameter is `Peter`.

```
public class LinkedList
{
   . . .
   public void addFirst(Object element)
   {
      Node newNode = new Node(); 1 2

      newNode.data = element; 3

      newNode.next = first; 4

      first = newNode; 5
   }
   . . .
}
```

Figure 5 - Object Diagram Interactive Exercise

The rearrange code exercise or Parsons problem is like a puzzle where the student rearranges the statements in correct order to get full score as depicted in Figure 6.


Rearrange the following lines to produce a program fragment that computes the area of a wall with two windows. First place all variable declarations, then the assignment statements.

```
double wallHeight = 8;
double windowHeight = 4;
int windows = 2;
double windowWidth = 3;
double wallArea;
wallArea = wallWidth * wallHeight;
double windowArea;
windowArea = windows * windowWidth * windowHeight;
wallArea = wallArea - windowArea;
double wallWidth = 30;
```

Figure 6 - Rearrange Code Interactive Exercise

3. Learning Tool Interoperability

3.1 Learning Management Systems

A **learning management system (LMS)** is a software application or platform used in educational institutions to deliver educational courses or training programs. It provides the facility for administrators and the instructor of a course to provide lecture materials, tests, quizzes, and assignments to the students, and keep track of their submissions and progress.

There are many existing LMS platforms used across educational institutions as well as online teaching , such as Canvas, Moodle, Coursera, Udacity, or edX. LMS has the facility to add lecture videos, multiple choice quizzes, and so on, but there is a need to integrate third party applications. For example not every learning management system run programming assignments. Common specifications are necessary to develop one single app that can be integrated with all the LMSes. These specifications are known as LTI specifications and the tools that follow these specifications are known as LTI tools.

Right now, there is no LTI tool that can integrate multiple CodeCheck or interactive exercises in any LMS. If the instructor wants to add multiple problems, he must create multiple assignments. This results in a large number of columns in the LMS gradebook.

3.2 LTI tools

LTI stands for Learning Tool Interoperability. LTI are specifications from IMS Global Learning Consortium, a nonprofit organization whose mission is to increase innovative learning technology. Its open architecture and products help education institutions to innovate and reduce the cost of integrating external applications into their enterprise systems used widely by teachers and students. There are many LTI tools available like coding practice platforms, recorded lecture videos for many subjects, multiple choice questions for chemistry, physics, history, languages, computer science etc.

LTI specifications are designed to enable easy integration of LTI tools with all the standard LMSes. Before LTI to connect a learning tool with a campus platform or LMS, the university had to take help from a developer or specialist to add the tool with the platform. This tool integration process might take months to complete and it needed to be done individually for each tool. The university might have to redo the complete process when the tool's new version became available. But if the tools follow the LTI specifications, it is comparatively easy to integrate the tool with the platform. LTI-compliant tools' integration with any LMS requires less time than what it would require if no standards are followed.

3.3 Key actors in LTI Systems

Two key actors in LTI systems are Tool Consumer and Tool Provider.

1. Tool Consumer :- The Learning Management System like Canvas or Moodle into which links to external tools are added.
2. Tool Provider :- The system providing access to one or more LTI Tools.

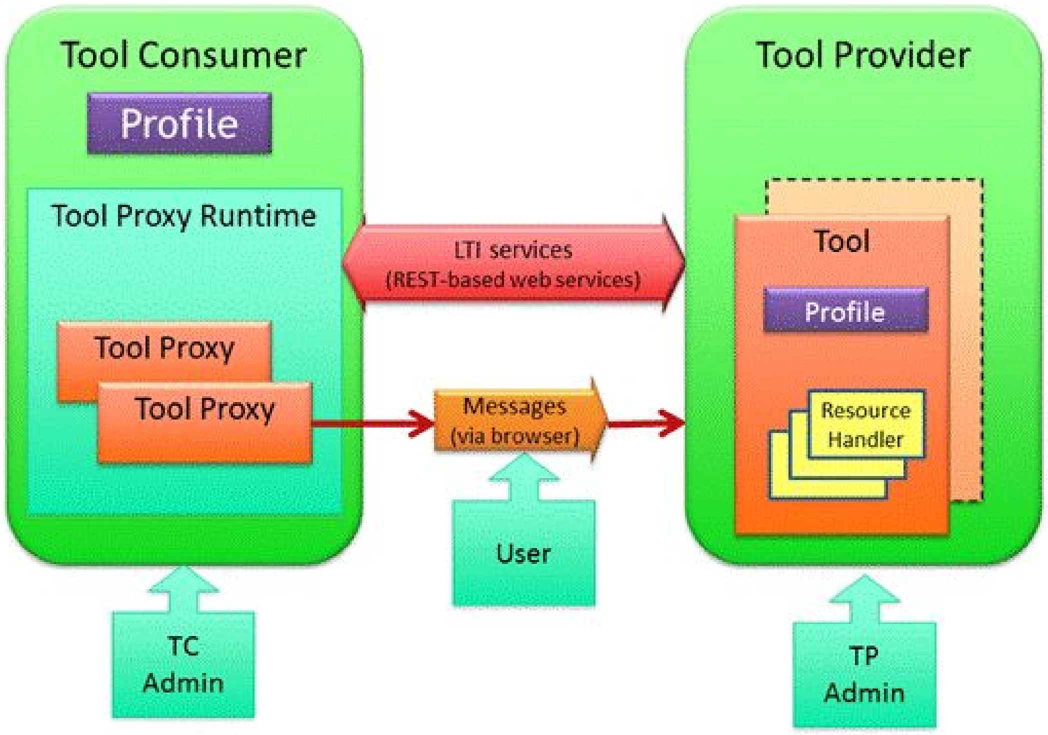


Figure 7 - TC and TP connection

The relationship between a Tool Consumer (TC) and a Tool Provider (TP) established via LTI is that the TC is responsible to authenticate and authorize users. The TC will provide TP with data about user, a user's current context(from where the LTI tool is launched i.e. which course or group within that course) and user's role within that context. This data is provided in a secure manner so that the TP may trust its authenticity. User in the Figure 7 is the person who is logged into LMS and launches the LTI tool. Messages refer to the data being transferred as a signed HTTP POST request via browser.

OAuth is an open standard for authorization used by Internet users to authorize websites or applications to access their information on other websites without giving them passwords. OAuth secures the data sent between any LMS and LTI app.

The LTI app provider should provide the LMS with a consumer key and secret key. The consumer key is a unique alphanumeric key assigned to each Tool Consumer by Tool Provider. The secret key is used to sign the messages. The LTI app provider uses consumer key to identify the Tool Consumer and find the associated secret key for validation. The consumer key and shared secret are added to the LMS when the LTI app is configured in the LMS for the first time. But some LMSes like Moodle provide the facility to add key and secret value every time you set up an assignment using any LTI app.

There are many ways to configure an LTI app in LMS :-

1. Paste the configuration URL - The configuration URL specifies where the XML configuration file of the LTI app is present. The XML file contains launch_url to which the LTI Launch request is sent when LTI app is launched within LMS.
2. Paste the XML file provided by LTI app provider to configure the app in LMS.
3. Manual configuration

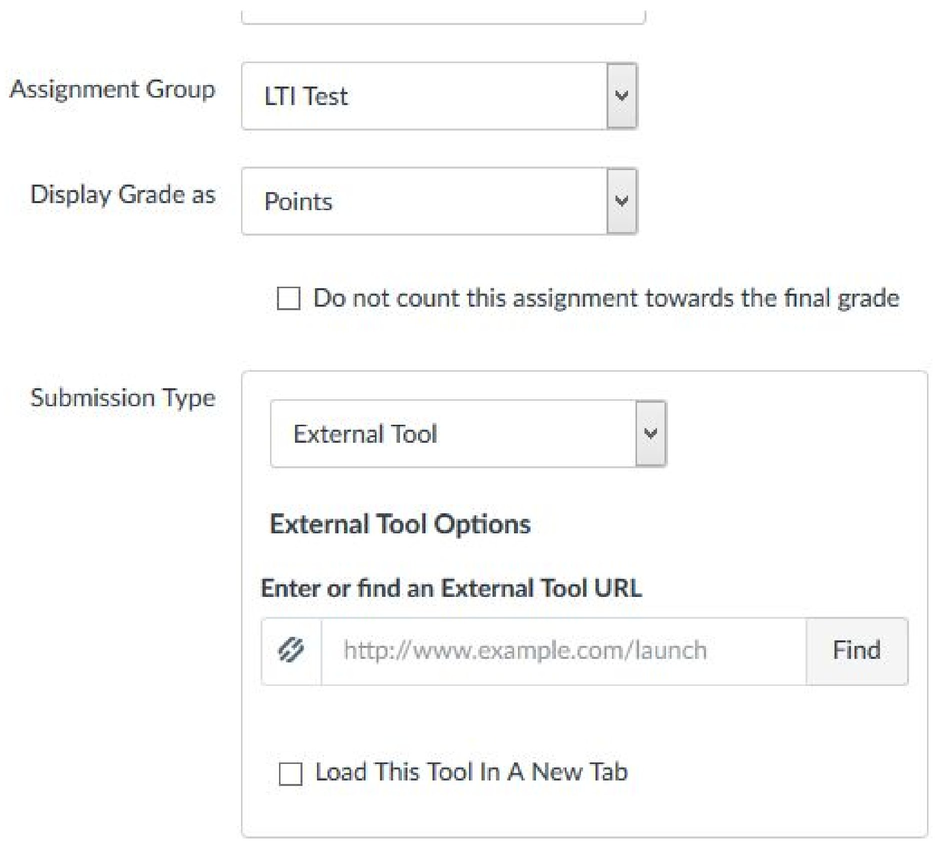
There are many advantages of LTI tools. First, there is no need for the student or the professor to create profiles or accounts. The LTI tool uses the user information sent from LMS and shows the appropriate landing page as per role. The student or professor doesn't go to any external system. The LTI tool is loaded within a LMS in an iframe(IFrame is an HTML page embedded within another HTML page) or in another window as per the instructor's settings. Many LTI tools return results or scores back to LMSes.

The main bottlenecks in the development of LTI applications is that there are no easy to follow specifications for LTI. and it’s an overhead for the developers to implement the LTI specifications for each and every tool that he wants to integrate with LMS. LTIHub integrates CodeCheck and certain interactive JavaScript elements, and the integration mechanism that I have used for the latter can easily be extended to other interactive elements.

4. Interaction Between LMS and LTI

4.1 Interaction between LMS and LTI in Instructor View

The instructor or administrator can add a third party LTI app to his course. To include the added LTI tool in an assignment, the instructor chooses Submission Type as “External Tool”. A dialogue box opens either to enter the external tool URL or find a specific tool from a list of third party tools by clicking on “Find” button as shown below:



Assignment Group: LTI Test

Display Grade as: Points

☐ Do not count this assignment towards the final grade

Submission Type: External Tool

External Tool Options

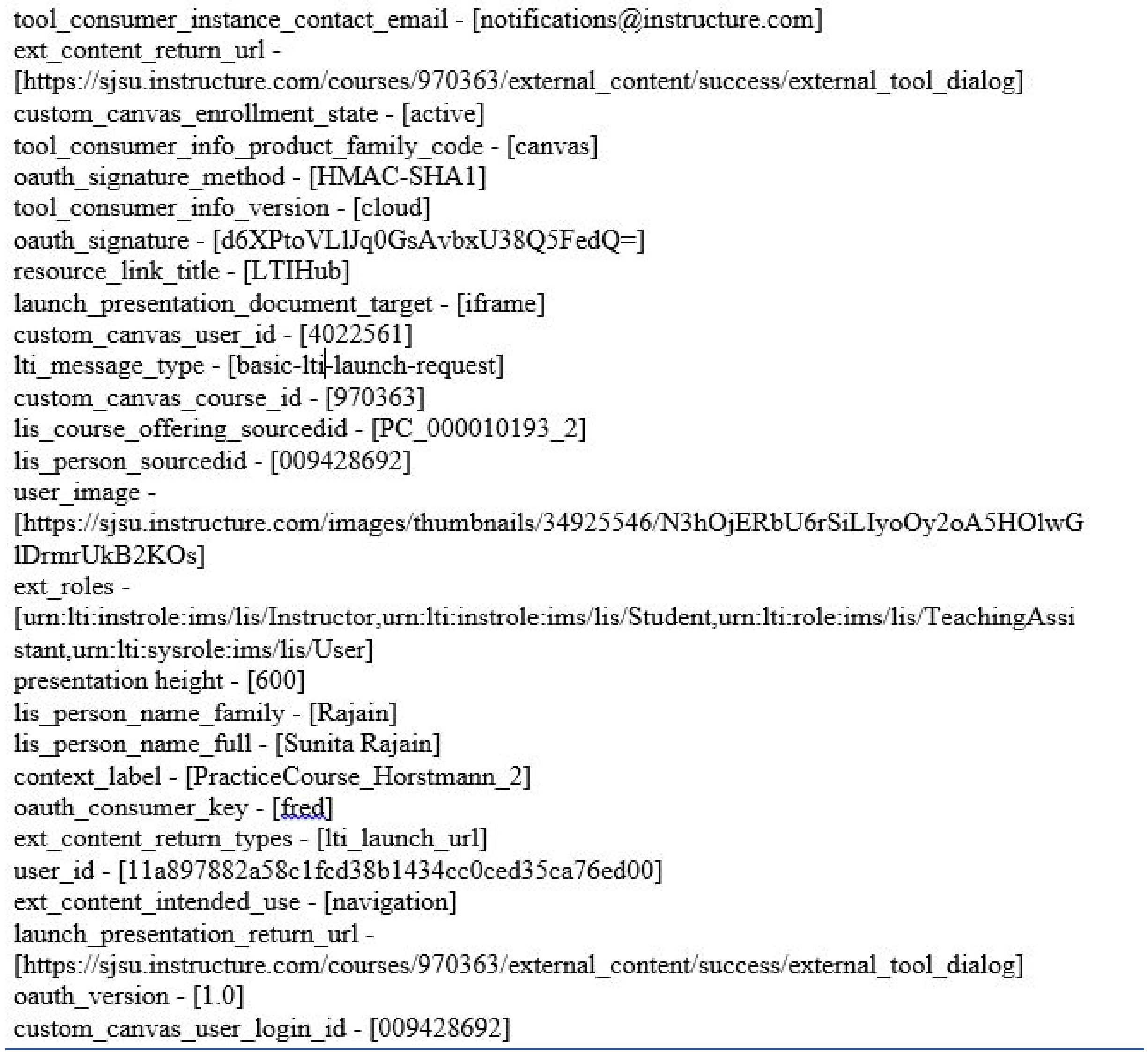
Enter or find an External Tool URL

http://www.example.com/launch

☐ Load This Tool In A New Tab

Figure 8 - Add External Tool to Assignment

An HTTP POST request is used for identity assertion from Tool Consumer to Tool Provider. When the instructor clicks on an LTI app link, a HTML form is filled with some pre-defined parameters. The form is submitted via JavaScript to an iframe rendered on a page within the Tool Consumer. Below is a list of some of the parameters that are sent from canvas LMS to LTIHub app server:

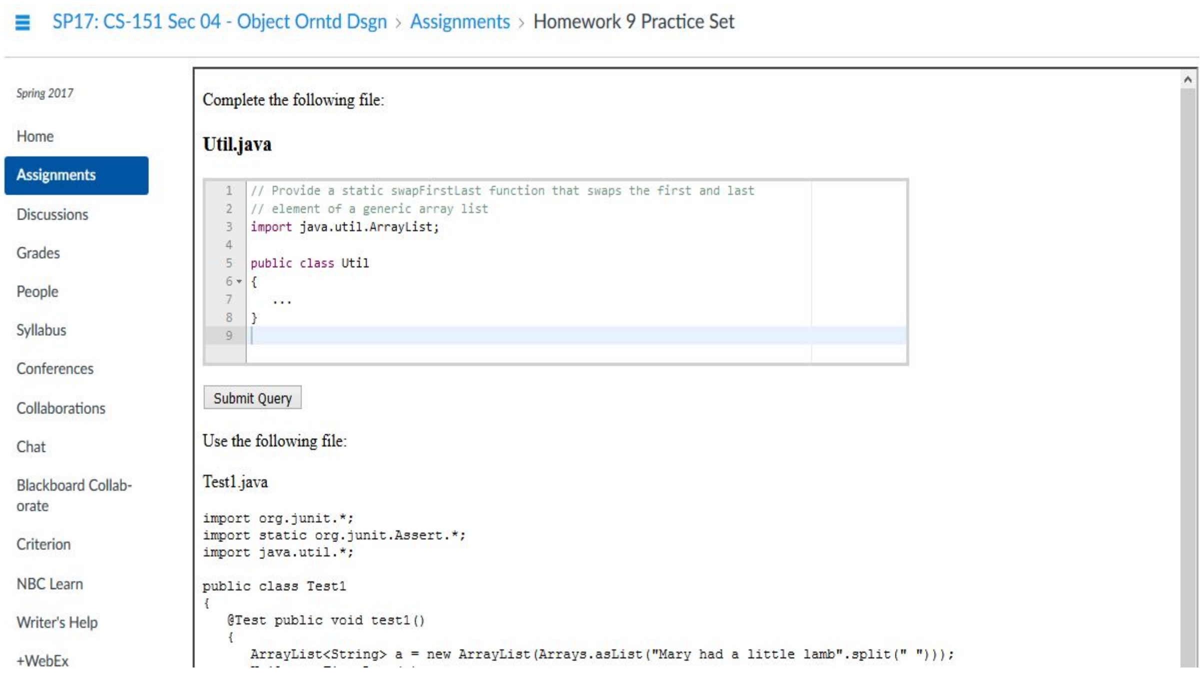


```
tool_consumer_instance_contact_email - [notifications@instructure.com]
ext_content_return_url -
[https://sjsu.instructure.com/courses/970363/external_content/success/external_tool_dialog]
custom_canvas_enrollment_state - [active]
tool_consumer_info_product_family_code - [canvas]
oauth_signature_method - [HMAC-SHA1]
tool_consumer_info_version - [cloud]
oauth_signature - [d6XPtoVL1Jq0GsAvbxU38Q5FedQ=]
resource_link_title - [LTIHub]
launch_presentation_document_target - [iframe]
custom_canvas_user_id - [4022561]
lti_message_type - [basic-lti-launch-request]
custom_canvas_course_id - [970363]
lis_course_offering_sourcedid - [PC_000010193_2]
lis_person_sourcedid - [009428692]
user_image -
[https://sjsu.instructure.com/images/thumbnails/34925546/N3hOjERbU6rSiLIyoOy2oA5HOlwG
lDrmrUkB2KOs]
ext_roles -
[urn:lti:instrole:ims/lis/Instructor,urn:lti:instrole:ims/lis/Student,urn:lti:role:ims/lis/TeachingAssi
stant,urn:lti:sysrole:ims/lis/User]
presentation height - [600]
lis_person_name_family - [Rajain]
lis_person_name_full - [Sunita Rajain]
context_label - [PracticeCourse_Horstmann_2]
oauth_consumer_key - [fred]
ext_content_return_types - [lti_launch_url]
user_id - [11a897882a58c1fcd38b1434cc0ced35ca76ed00]
ext_content_intended_use - [navigation]
launch_presentation_return_url -
[https://sjsu.instructure.com/courses/970363/external_content/success/external_tool_dialog]
oauth_version - [1.0]
custom_canvas_user_login_id - [009428692]
```

Figure 9 - Parameters Sent from LMS to LTI

4.2 Interaction between LMS and LTI in Student View

When a student logins into LMS, the created assignment will be available for him to do. Clicking on assignment link will load the assignment. The figure below shows an assignment in Canvas that is created using LTIHub app.



Complete the following file:

Util.java

```
// Provide a static swapFirstLast function that swaps the first and last
// element of a generic array list
import java.util.ArrayList;

public class Util
{
   ...
}
```

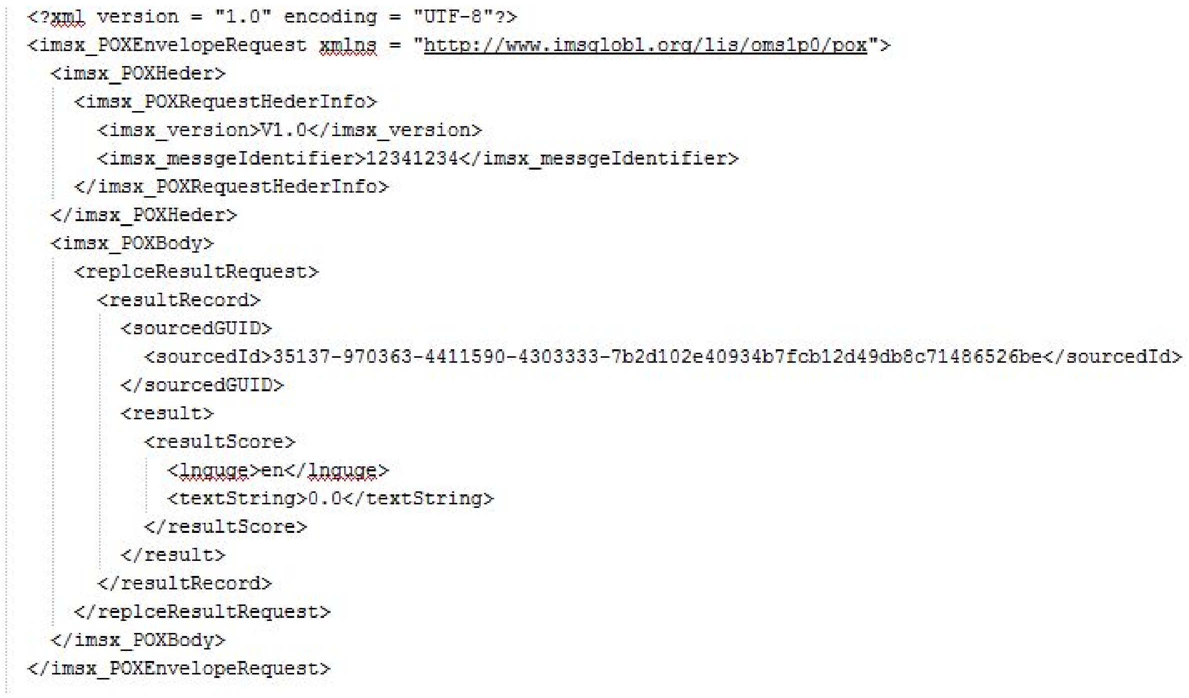
Use the following file:

Test1.java

```
import org.junit.*;
import static org.junit.Assert.*;
import java.util.*;

public class Test1
{
   @Test public void test1()
   {
      ArrayList<String> a = new ArrayList(Arrays.asList("Mary had a little lamb".split(" ")));
```

Figure 10 - Assignment as Shown in Student View

If the LTI app is supposed to send the grades back to canvas, it is sent in a XML file as specified by IMS. Below is a sample XML file sent from LTIHub to LMS:



```
<?xml version = "1.0" encoding = "UTF-8"?>
<imsx_POXEnvelopeRequest xmlns = "http://www.imsglobl.org/lis/oms1p0/pox">
  <imsx_POXHeder>
    <imsx_POXRequestHederInfo>
      <imsx_version>V1.0</imsx_version>
      <imsx_messgeIdentifier>12341234</imsx_messgeIdentifier>
    </imsx_POXRequestHederInfo>
  </imsx_POXHeder>
  <imsx_POXBody>
    <replceResultRequest>
      <resultRecord>
        <sourcedGUID>
          <sourcedId>35137-970363-4411590-4303333-7b2d102e40934b7fcb12d49db8c71486526be</sourcedId>
        </sourcedGUID>
        <result>
          <resultScore>
            <lnguge>en</lnguge>
            <textString>0.0</textString>
          </resultScore>
        </result>
      </resultRecord>
    </replceResultRequest>
  </imsx_POXBody>
</imsx_POXEnvelopeRequest>
```

Figure 11 - XML Grade File Sent from LTIHub to LMS

5. System Architecture

I have developed a system called LTIHub that follows the LTI specifications. LTIHub acts as a mediator between any LMS and Learning Tool Provider. The Learning Tool Provider needs to provide URL for each problem and a callback URL. The system allows instructor to add multiple CodeCheck or interactive problems in a single assignment. This intermediator application stores the information related to the status of assignments and pass the grades back to the LMS when all the assignment's problems are done by the student.

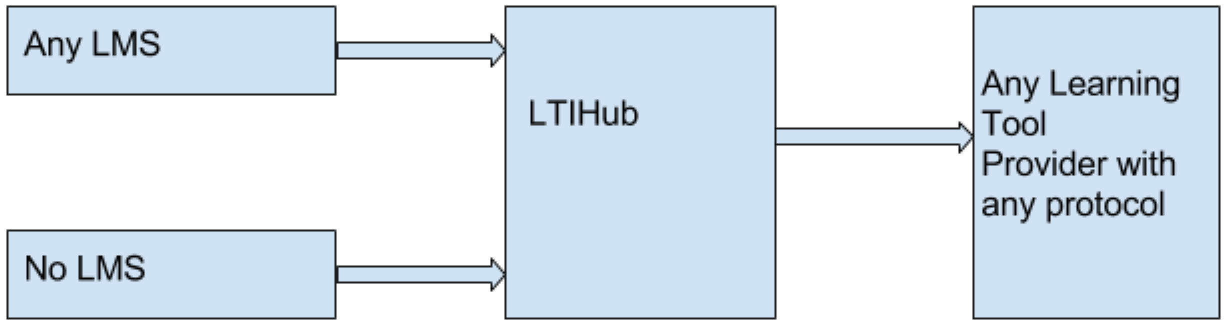


Figure 12 - LTIHub Architecture

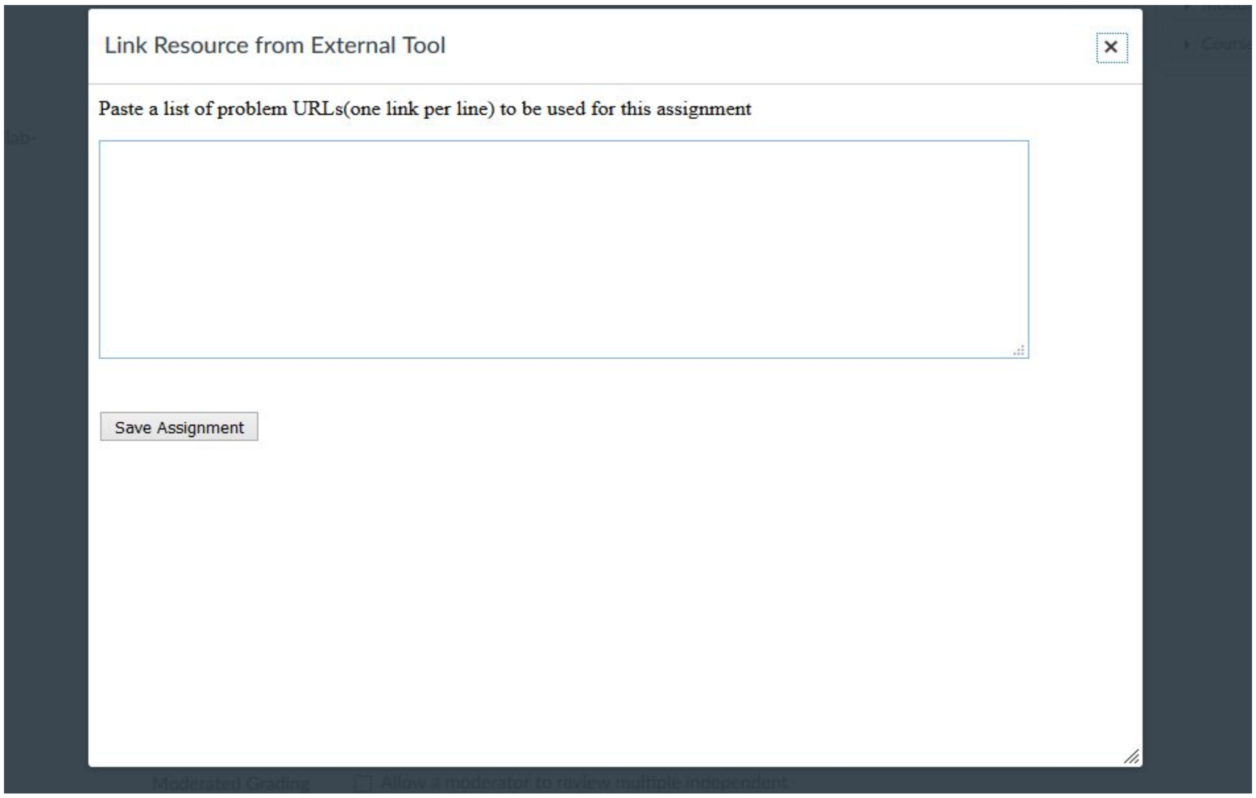
The system designed has been tested with Wiley interactive exercises and CodeCheck programming problems in Canvas LMS. It can be used to add potentially other activities into LMS.

I used an AWS EC2 instance and a MySQL database during the implementation phase and a Google Compute Engine VM and Google Cloud SQL during the production phase. As the communication between any LMS and LTI is done over HTTPS only and AWS provides SSL certification at ELB(Elastic Load Balancing) level, I used ELB to delegate the incoming requests to the Amazon instance.

I used the Play Framework to write the code for the web application which is a MVC based framework available for both Java and Scala. There are three entities in my app: Assignment, Problem and Submission. Assignment has one-to-many associations with Problem. Problem has one-to-many associations with Submission.

6. Communication Between LTIHub and LMS

LTIHub can be added to any course by pasting the app configuration XML file. When the instructor wants to include LTIHub in a assignment, he clicks on LTIHub link from the list of external tools and the pre-defined parameters are passed from LMS to LTIHub in a POST request. The LTIHub checks the value of ext_role, lis_outcome_service_url, lis_result_sourcedid, launch_presentation_return_url parameters. The lis_outcome_service_url and lis_result_sourcedid parameters are sent from LMS to LTI when a student(i.e. ext_role value is either student or learner) accesses the LTI tool and the tool is supposed to pass back the grades to LMS. If ext_role value is either "Faculty","Teaching Assistant" or "Instructor", the assignment creation page is opened as shown below :



Paste a list of problem URLs(one link per line) to be used for this assignment

Figure 13 - LTIHub Assignment Creation Page

The instructor adds the one problem URL per line and clicks on “Save Assignment” button. The next window shows the added problems list and two buttons - “Done” and “Edit”. If the instructor is ok with the assignment, he clicks on “Done” button and a unique URL pointing to that assignment is generated and sent back to the LMS. If the instructor wants to add more problems or delete an added problem, he clicks on “Edit” button. The “launch_presentation_return_url” parameter specifies the path where the generated assignment URL should be sent. The JavaScript code used to generate the assignment URL so that it is not tied to one domain is shown below:

```
<script type="text/javascript">
  function modifyURL()
  {
    var ltiToolUrl = window.location.protocol + "//" +
    window.location.host + "@preFix" + "/assignment?id=";
    document.myform.url.value = ltiToolUrl +
    encodeURIComponent(document.myform.url.value);
    return true;
  }
</script>
```

The created assignment is shown in instructor login as it would be visible in “Student Login View” with one dialog box saying that “The assignment is shown in the instructor view” and a “Edit Assignment” button is also available at the bottom of the assignment. The instructor can add or remove problem URLs from an assignment even after the assignment has been added to the LMS.

When student logs into the LMS, the assignment created by the instructor will be available for him to do. Clicking on the assignment will load the LTIHub tool in the same iframe or another window as per the setting chosen by the instructor. There are two ways in which a student's submission is graded by the third-party app:-

1. Student's submission is sent to the third-party server as in the case with CodeCheck problem
2. Student's response is checked in the browser window itself by a JavaScript code as in the case with interactive exercises.

I wrote two methods to take care of saving student's score for both types of exercise problems. For CodeCheck problems, I wrote a JavaScript code that runs when the window loads and appends a callback URL to each CodeCheck problem. The code is shown below.

```
function onLoadHandler() {
                           var              iframes              =
document.getElementsByClassName('exercise-iframe')
  for (i = 0; i < iframes.length; i++) {
    var str = iframes[i].src;
    var patt = new RegExp("play.codecheck.ws");
    if(patt.test(str) && str.includes("?") ){
      iframes[i].src = iframes[i].src + "&scoreCallback =" +
      window.location.protocol + "//" + window.location.host
      +  "@preFix" + "/submissions/" + "@assignmentID"
       + "/"  +  "@userID"  + "/" + iframes[i].id;
    }
    else if(patt.test(str)){
      iframes[i].src = iframes[i].src + "?scoreCallback =" +
      window.location.protocol +"//" + window.location.host
      + "@preFix" + "/submissions/" + "@assignmentID" + "/"
      + "@userID" + "/" + iframes[i].id;
    }
    else
     iframes[i].src = iframes[i].src;
  }
}
```

When the student submits solution for CodeCheck problem, the complete file is sent to CodeCheck server for evaluation. After evaluation, the report sent by CodeCheck server is shown to the student as well as sent to controllers method as specified by the callback URL. The report is sent in JSON format.

For interactive exercises, nothing is sent from LMS to the third party server. Everything happens in the browser itself. To save the student's score for this exercise, I wrote JavaScript code to use the Window.postMessage() method which makes it possible to pass messages across different domain. There is no other way of knowing whether the student attempted any problem or not because there is no communication with any server. Everything is controlled by the JavaScript running in the browser window. I used the setInterval() method to invoke postMessage() method to iframes to check if there is any problem solved by a student whose score should be saved in the databases. The code is shown below.

```
<script type="text/javascript">
  var exerciseScores = [];
  function checkScores() {
    exerciseScores = [];
    $('.exercise-iframe').each(function (_, iframe) {
      iframe.contentWindow.postMessage("scores", "*")
    });
    return false
  }

  function receiveMessage(event) {
                                    var         iframes         =
document.getElementsByClassName('exercise-iframe')
    exerciseScores.push(event.data)
    if (exerciseScores.length === iframes.length) {
     $.ajax({
     Url: '@preFix/send-interactivescore/@assignmentID',
      method: 'POST',
      contentType: 'application/json',
      data: JSON.stringify(exerciseScores),
      success: function(msg) {
        var response = document.getElementById('response')
        response.innerHTML
        = "Your partially complete assignment is saved!";
         setTimeout(function() { $("#response").hide(); }, 4000);
        }
      });
    }
  }
  window.addEventListener("message", receiveMessage, false);
  setInterval(checkScores,60000);
</script>
```

The student responses are saved so it's ok for the student to attempt some problems and then take a break. The last attempted highest score for each problem will be shown when the assignment is loaded again. If the students want to improve their grades then they can attempt as many times as they want. To test how strong the students are in their coding, there is also a facility to assign timed assignment and to check the student's performance in the time-constrainted exam.

After the student has done all the CodeCheck and interactive exercises given in a particular assignment, the grades are sent to LMS from LTIHub. The grades are sent in XML file from LTIHub to the LMS. The XML file has a resourcedId that identifies the place where grades are to be sent in LMS.

7. Experiments and Results

The system was deployed in Spring 2017 in four classes at SJSU. In the following sections, I describe about the students performance in terms of average time and number of submissions as well as other experimental result.

7.1 CMPE180-92

CMPE180-92 - Data Structures and Algorithms using C++ is a refresher course for students who failed an entrance exam and students who have marginal programming skills. There were 105 students in this class. 10 practice exercises were offered in this class. Students earned points for doing the practice problems that counted towards their final grades.

To find out how this app helped in students' learning, I analyzed the student submissions based on two criteria: Numbers of submissions per problem and time spent working on a problem.

Figure 14 shows the average time worked for all problems. To calculate the average time, we took the time stamp of all submissions of a problem. We added the time difference between submissions of a particular problem by a student if the difference is less than 20 minutes. There were 105 students enrolled in the course out of which more than 75 students were considered for average time difference calculation. The student who made only single submission were excluded. The average time calculated is error prone as we have no information about the time that the student spent when he tried the problem for the first time.

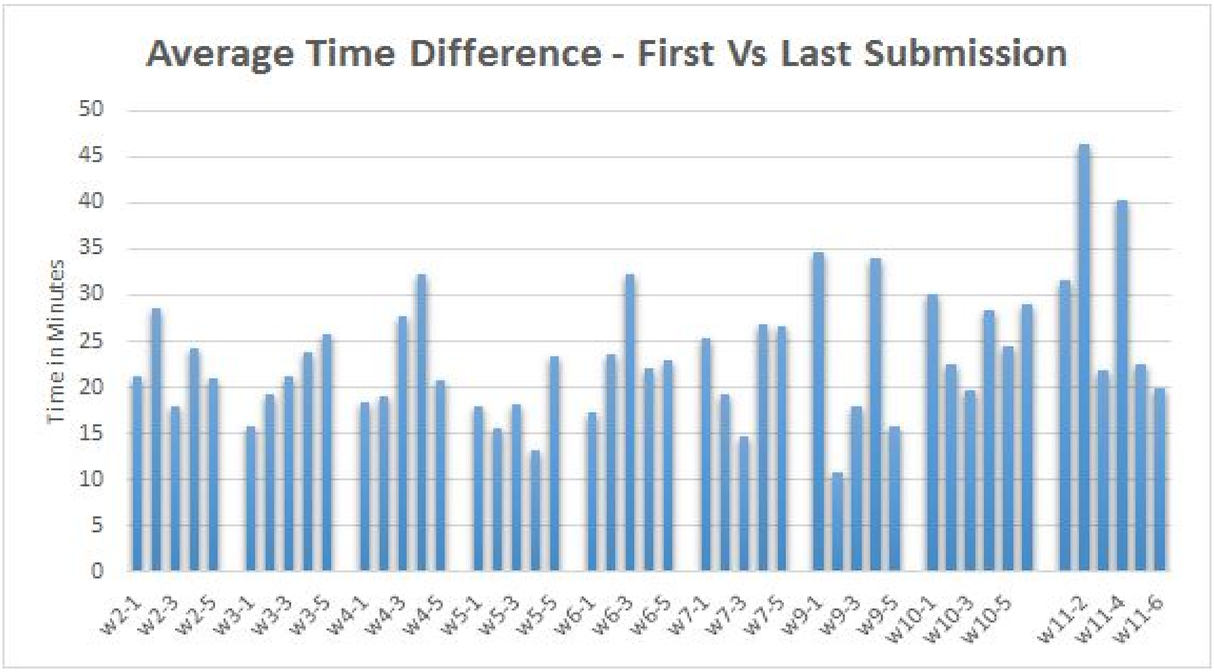


Figure 14 - Average Time Difference in Submissions for CMPE180-92 Students

As evident from the above graph, the time difference is highly variable and does not show obvious improvement. Variations in the graph might be attributed to the topics being hard for the students to understand. The topics for each week were picked to match weekly homework assignments:

1. Input and output (Week 1 was dropped from the analysis because of technical difficulties.)
2. Functions
3. Arrays
4. Pointers
5. Classes
6. Copy constructors
7. Implementing linked lists
8. (No exercises)
9. Using STL lists and iterators
10. Recursion

11. Maps

Figure 15 shows the total number of attempts for each problem.

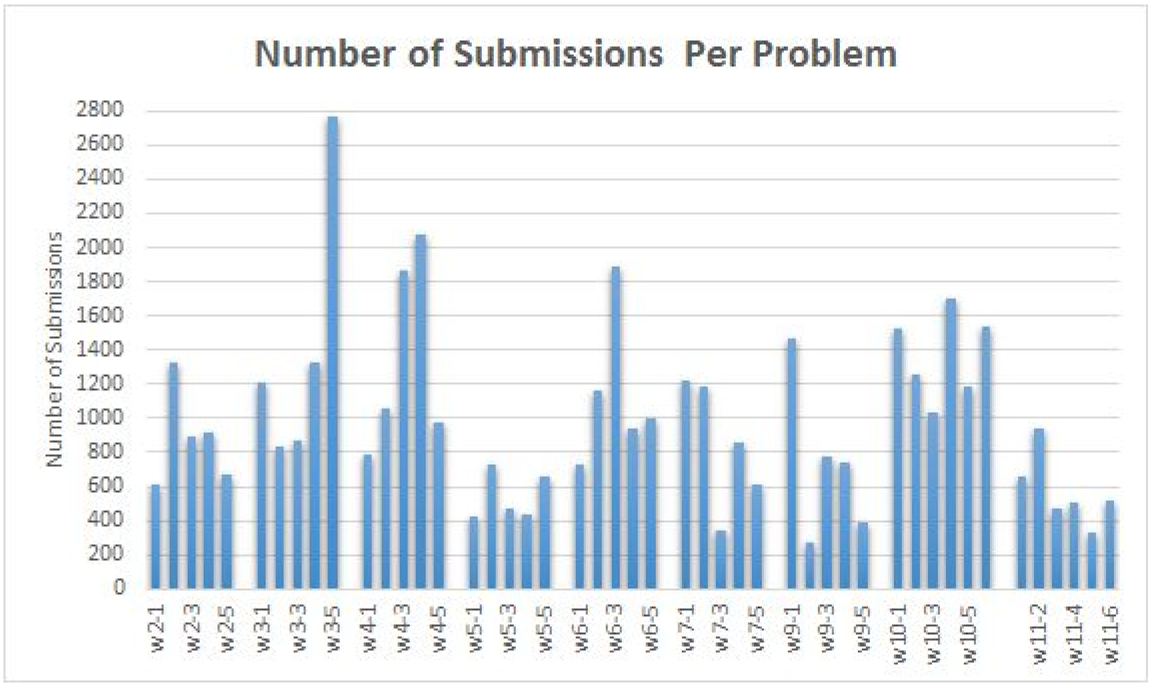


Figure 15 - Total Number of Submissions Per Problem for CMPE180-92 Students

Difficulty levels of these problems were determined separately by Professor Mak and Professor Horstmann. More than 50% problems have the same difficulty level assigned by both professors. 40% differ by 1 point and 10% differ by 2 points. I took the average of difficulty levels assigned by the professors (Figure 16). Correlation coefficient between number of submissions and difficulty level of problem is 0.264. The correlation coefficient between average time between first and last submission vs difficulty level of problem is 0.29. I didn't calculate correlation between practice problem scores and difficulty level as eventually most students got full marks. It's evident that there is no strong correlation between difficulty level of problems as assigned by instructors and number of submissions and average time spent per problem by the student. I conclude that there is a difference in perception of difficulty level of a topic between instructors and students.

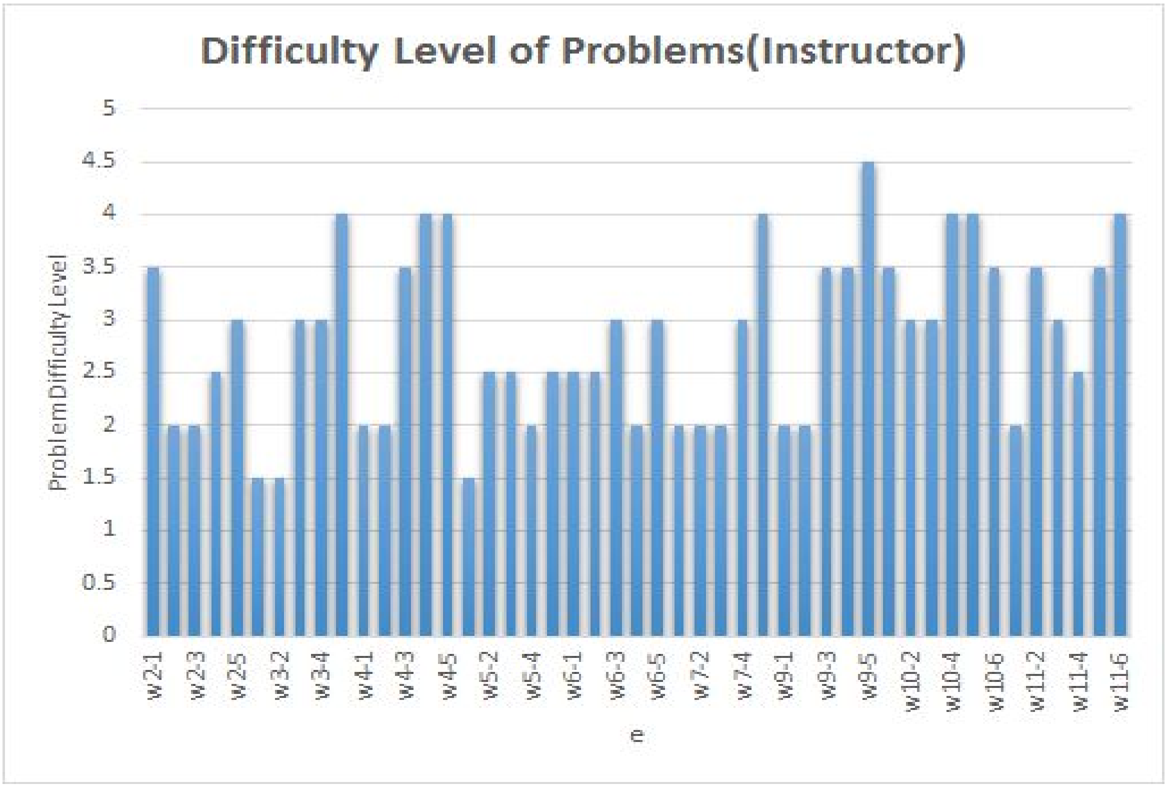


Figure 16 - Difficulty Level of Problems Given to CMPE180-92 Students

To find out whether these exercises helped the students understand the topics better and perform better in exams, I analyzed the scores of students in practice exercises and mid term exam. I computed the linear correlation between students' scores in practice exercises and the midterm exam as shown in Figure 17. I also computed the linear correlation between scores in practice exercises and homework assignments (Figure 18).

The linear correlation between practice tests and exam scores is low at 0.17. The linear correlation coefficient increases to 0.29, if we don't consider scores of 9 with low participation in practice tests that are circled in the Figure 17.

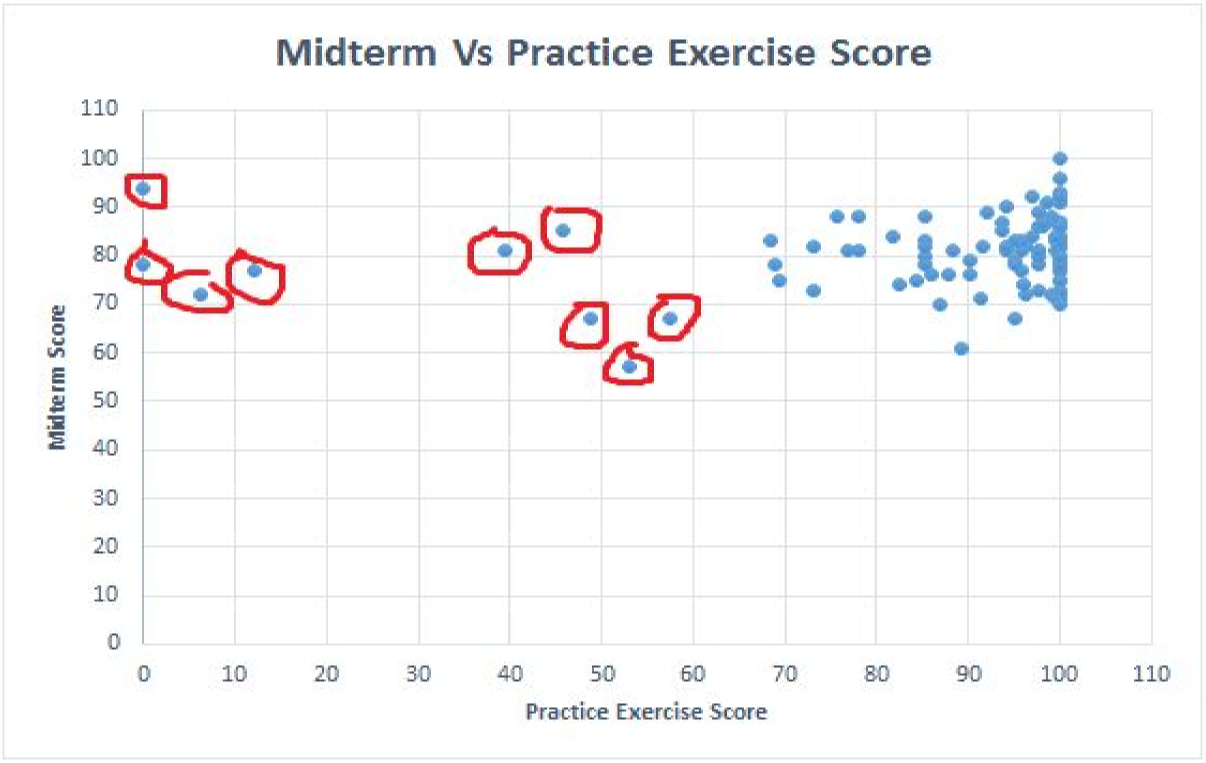


Figure 17 - Midterm Vs Practice Exercises Score for CMPE180-92 Students

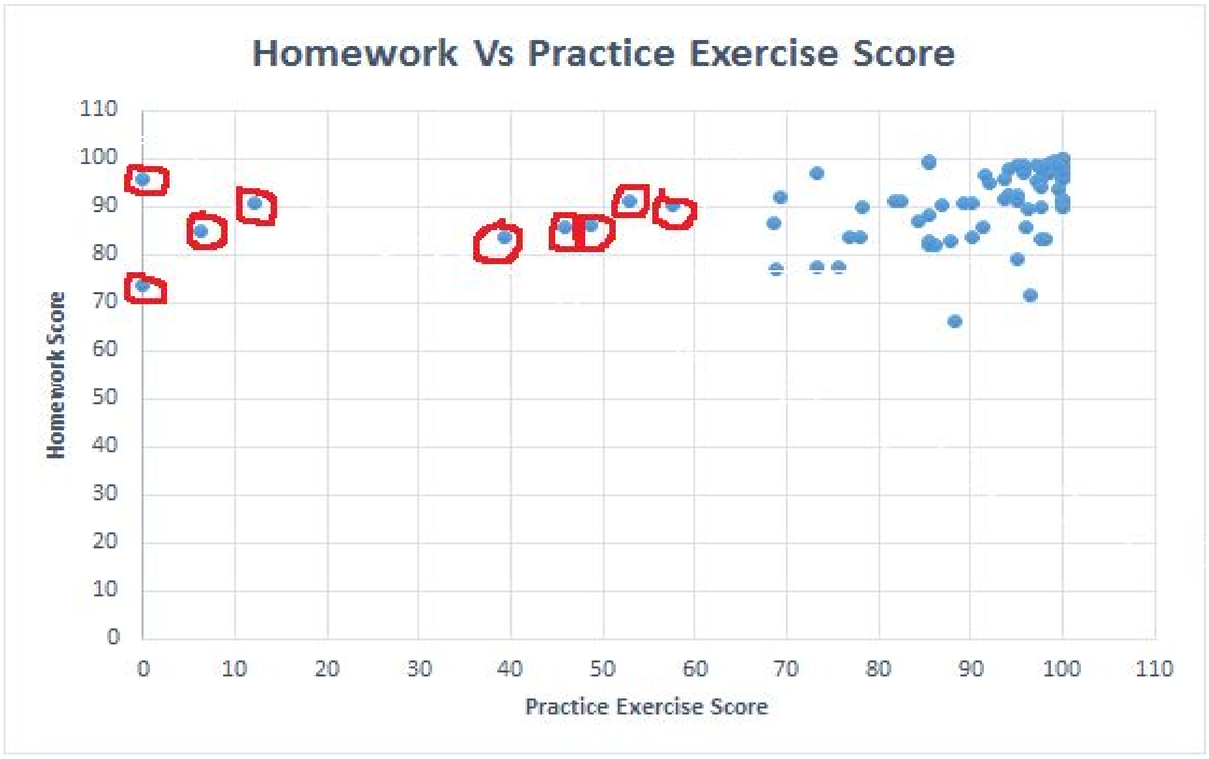


Figure 18 - Homework Vs Practice Exercises Score for CMPE180-92 Students

The linear correlation coefficient between homework and practice exercises score is 0.42 and increases to 0.56, if we don't consider the 9 students that are circled in Figure 18.

In students survey[B.2], 98.4% students said that these practice exercises helped them to prepare for homework and exams but correlation coefficient value doesn't indicate that practice tests and exam scores are related, but practice test performance might be related to homework scores.

Looking at the students, I hypothesize that they fall into three groups. Students with good programming knowledge will not improve with practice exercises. Students who are not active participants will also not improve. I wanted to follow the third group of students that were struggling at first but participating regularly, and I wanted to know how their scores were changing as the semester progressed. I looked at the students' scores in the first and second week. I took out the students who were getting full scores and the students who were not participating. Then I was left with a group of 18 students who got very low score in week 1 and week 2 practice exercises. I found that out of those 18 students, 12 students scores were showing improvement week by week and their midterm scores were above class average. I don't have any proof that this improvement was because of these practice exercises, but some enhanced experimentation can be done in the future to find out.

7.2 CS151

Two practice exercises were done in this class. The first practice exercise involved questions (labelled from p7-1 to p7-6 in the Figure 19) from inheritance that were very basic. The second practice exercise problems were based on Generics and wildcard that is a comparatively hard topic as evident in the average time graph. Each practice set had one pretest and one posttest. Unlike the practice exercises the pretest and posttest were timed (30 minutes). The pretest assignment was given to analyze the student's understanding of the topic. The posttest was used to see if the practice exercises have helped the students to perform better in the posttest.

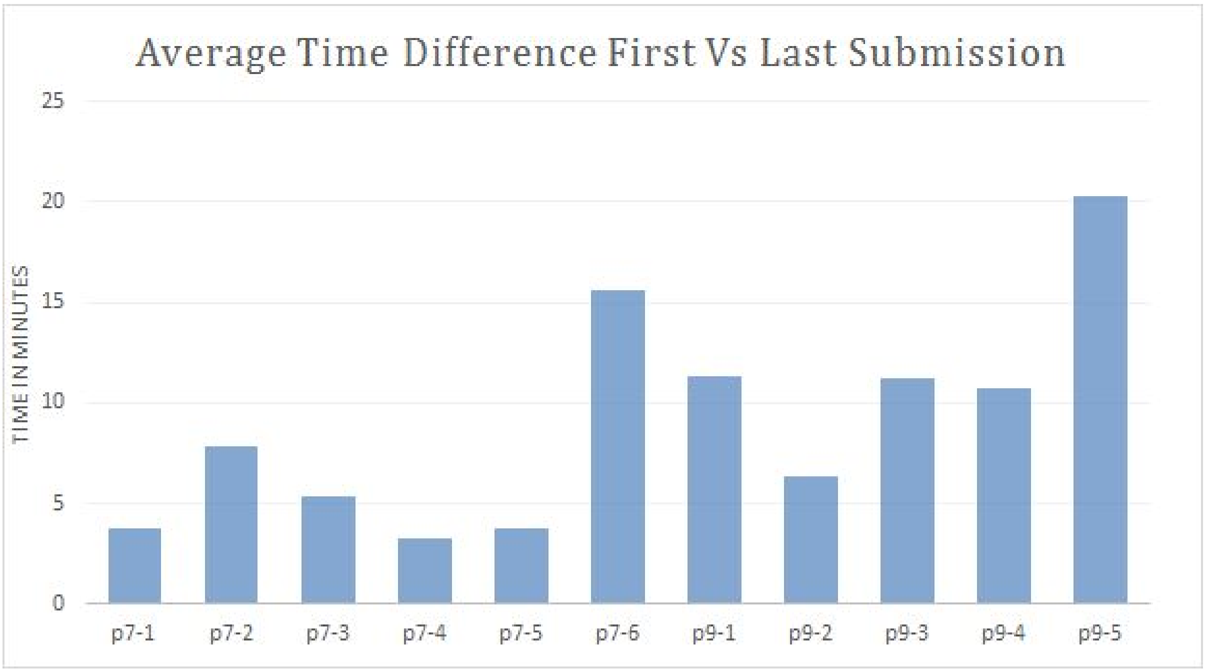


Figure 19 - Average Time Difference in Submissions for CS151 Students

Pre- and post-test experiment requires more thought as it is more complex to set up than originally thought.

7.3 CS156 and CS149

CS156 is a class in artificial intelligence where the instructor uses Python, which is not a prerequisite. There are two sections in this class. Section 1 has 31 students and section 2 has 29 students. Two practice exercises were done in each section. The instructor declined the offer to add more exercises because of low student participation. Only 14 students from section 1 and 5 students from section 2 participated practice exercise 1. Practice exercise 2 was done by only 4 students from section 1 and 3 students from section 2. As there was negligible participation, we didn't get any informative analysis from this class.

CS149 is an operating systems class where the instructor uses C, which is not a prerequisite. There were two sections but only one section participated since I hoped to compare the sections with and without the exercises. Two practice exercises were offered in this class. There are 70 students enrolled in the class but grades in canvas shows that only 17 students got scores for first practice exercise and only 5 students got scores for second practice exercise. The instructor expressed that the practice exercises were useful but as his homework moved on to more complex operating systems topics, it became too challenging to provide practice exercises that were tailored to the course contents. As there was very little participation, we didn’t get any informative analysis from this class.

Both instructors gave no points for the practice exercises. If one were to repeat this in the future, it would be important to make both the instructors and students understand about the values and benefits of such exercises so that we can have better participation.

8. Conclusions

Integration works fine as assignment were created successfully for both CodeCheck and interactive exercises. Grades were successfully passed from LTIHub to canvas but more work is needed on persisting student work. Right now, if the student takes a break and then reloads the assignment, last best achieved score for each problem is shown to him. If would be good if the student's work/submitted code is also shown as it would help student to analyze if he is ok with the approach taken or he wants to try another approach or logic.

There seems to be some correlation between practice and homework/exam as depicted by CMPE180-92 students scores graph in Figure 17 and Figure 18 but it's not very strong. Pre- and post-test experiment requires more thought as it is more complex to set up than originally thought.

Near about 60% students from CMPE180-92 reported that they have spent between 4 to 6 hours or more than 6 hours per week for the practice exercises as shown in Student Survey in Appendix B.1 but the time spent is between 1.5 to 2.5 hours as shown in Average Time graph in Figure 14 which is completely in disagreement with student perceptions. But most students copied the problems into Eclipse and solved them there before pasting their solutions back into CodeCheck. We don't know how much time or how many attempts they made in Eclipse. We compared results/effort against perceived the instructor difficulties and found that there is a difference in perception of difficulty of a topic between instructors and students.

We analyzed students submissions for plagiarism and found that there is some amount of cheating going on, even though the reward given is very low. Some students cheat after having given up on their approach. It would be useful in the future to explicitly analyze the attempts before the successful submission. Seeing multiple submissions is a very valuable resource for investigating plagiarism. A future version of this tool should make it much easier for instructors to explore this.

Students liked the concept of these practice exercises as evident from CS149 students survey available in Appendix B.2. 22 students took part in survey and 86% students claimed that these practice exercises are helpful for their homework/ exam preparation as well as they would like to recommend it to other students and would like these practice exercises in their other Programming courses also. It sounds interesting as only 17 students have participated in the practice exercises. It looks like students don't like to participate for such experimentation if there are no points for it. Students received points in CMPE 180-92 for the practice exercises so the students participated actively in all the practice exercises.

Looking at the students' survey, we can see that students consider these practice exercises helpful for their homeworks as well as exams preparation. The students would like to have these exercises in their other courses also. It seems that the LTIHub is useful for the students. With the submission analysis feature addition and the last best solution availability, usability of these exercises can be enhanced.

9. References

[Ahadi 2016] Alireza Ahadi, Raymond Lister, and Arto Vihavainen. 2016. On the Number of Attempts Students Made on Some Online Programming Exercises During Semester and their Subsequent Performance on Final Exam Questions. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE '16). ACM, New York, NY, USA, 218-223. DOI: http://dx.doi.org/10.1145/2899415.2899452

[Edgcomb 2017] Alex Edgcomb, Frank Vahid, Roman Lysecky, and Susan Lysecky. 2017. Getting Students to Earnestly Do Reading, Studying, and Homework in an Introductory Programming Class. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, USA, 171-176. DOI: https://doi.org/10.1145/3017680.3017732

[Edwards 2001] Stephen H. Edwards and Manuel A. Perez-Quinones. 2008. Web-CAT: automatically grading programming assignments. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education* (ITiCSE '08). ACM, New York, NY, USA, 328-328. DOI=http://dx.doi.org/10.1145/1384271.1384371

[Edwards 2008] Stephen H. Edwards, Jürgen Börstler, Lillian N. Cassel, Mark S. Hall, and Joseph Hollingsworth. 2008. Developing a common format for sharing programming assignments. *SIGCSE Bull.* 40, 4 (November 2008), 167-182. DOI= http://dx.doi.org/10.1145/1473195.1473240

[Jurado 2014] F. Jurado and M. A. Redondo, "Learning tools interoperability for enhancing a distributed personal learning environment with support for

programming assignments," *2014 International Symposium on Computers in Education (SIIE)*, Logrono, 2014, pp. 87-92.
doi: 10.1109/SIIE.2014.7017710

[Moumoutzis 2014] G. Stylianakis, N. Moumoutzis, P. Arapi, M. Mylonakis and S. Christodoulakis, "COLearn and open discovery space portal alignment: A case of enriching open learning infrastructures with collaborative learning capabilities," *2014 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL2014)*, Thessaloniki, 2014, pp. 252-256.
doi: 10.1109/IMCTL.2014.7011142

[Lahtinen 2005] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. ‚bA study of the difficulties of novice programmers. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education* (ITiCSE '05). ACM, New York, NY, USA, 14-18. DOI=http://dx.doi.org/10.1145/1067445.1067453

[Norris 2008] Cindy Norris, Frank Barry, James B. Fenwick Jr., Kathryn Reid, and Josh Rountree. 2008. ClockIt: collecting quantitative data on how beginning software developers really work. *SIGCSE Bull.* 40, 3 (June 2008), 37-41. DOI=http://dx.doi.org/10.1145/1597849.1384284

[Pieterse 2013] Vreda Pieterse. 2013. Automated Assessment of Programming Assignments. In *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research* (CSERC '13), Marko van Eekelen, Erik Barendsen, Peter Sloep, and Gerrit van der Veer (Eds.). Open Universiteit, Heerlen, Open Univ., Heerlen, The Netherlands, The Netherlands, , Article 4 , 12 pages.

[Sheard 2011] Judy Sheard and Martin Dick. 2011. Computing student practices of cheating and plagiarism: a decade of change. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (ITiCSE '11). ACM, New York, NY, USA, 233-237. DOI=http://dx.doi.org/10.1145/1999747.1999813

[Simon 2013] Simon. 2013. Soloway's Rainfall Problem Has Become Harder. In *Proceedings of the 2013 Learning and Teaching in Computing and Engineering* (LATICE '13). IEEE Computer Society, Washington, DC, USA, 130-135. DOI=http://dx.doi.org/10.1109/LaTiCE.2013.44

[Thomas 2001] P. Thomas and C. Paine, "How students learn to program: observations of practical tasks completed," *Proceedings IEEE International Conference on Advanced Learning Technologies*, Madison, WI, 2001, pp. 170-173. doi: 10.1109/ICALT.2001.943891

[Vihavainen 2014] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual conference on International computing education research* (ICER '14). ACM, New York, NY, USA, 19-26. DOI: http://dx.doi.org/10.1145/2632320.2632349

[Zhenghao 2015] Zhenghao, Chen, Brandon Alcorn, Gayle Christensen, Nicolas Eriksson, Daphne Koller, and Ezekiel J. Emanuel. "Who's Benefitting from MOOCs, and Why"." *Harvard Business Review*. Harvard Business Publishing, 22 Sept. 2015. Web.

[Zingaro 2005] Daniel Zingaro, Andrew Petersen, and Michelle Craig. 2012. Stepping up to integrative questions on CS1 exams. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE

'12). ACM, New York, NY, USA, 253-258. DOI=http://dx.doi.org/10.1145/2157136.2157215

Cay Horstmann. CodeCheck. [Online]. http://horstmann.com/codecheck/

Cay Horstmann. CodeCheck. [Online]. http://horstmann.com/codecheck/authoring.html

Cay Horstmann. InterActivities. [Online]. http://horstmann/private/apps/worksheet.html

Ims Global. [Online]. https://www.imsglobal.org/activity/learning-tools-interoperability

OAuth-wikipedia. [online]. https://en.wikipedia.org/wiki/OAuth

Appendix

A. Implementation and System Set up

1. Steps to setup EC2 instance

- Sign up for AWS and click on “Launch Instance” button.
- Choose the AMI and then choose an Instance Type.
- Keep the default settings for Storage and click on “Review and Launch” button.
- Configure Security Group to open HTTP, HTTPS and SSH ports.
- Clicking on “Launch” button opens a popup window to create a new public- private key pair. Public key is stored by Amazon and private key is stored by user. Private key is used to connect securely to the AWS instance.

2. Required Software Installation and Setup

- SSH to your AWS instance using the private key and public IP of the instance.
- Check which java is installed on AWS instance using command
```
java -version
```
- By default, openjdk will be installed. Remove it and all the dependent file by using command
```
sudo apt-get purge openjdk-(version i.e. 7 or 8)-jre
openjdk-(version i.e. 7 or 8)-jre-headless
```
- We need Java 8 from sun microsystems for our program. To install sun java8, use command
```
sudo add-apt-repository ppa:webupd8team/java
sudo apt update
sudo apt install oracle-java8-installer
```
- Set the JAVA_HOME environment variable in the file /etc/environment.

Save and reload the file.

- Install apache server by command and run the service

```
    sudo apt-get install apache2
```

- Download Play framework zip file from https://www.typesafe.com/activator/download, unzip it and set ACTIVATOR_HOME path and export it using command

```
    export PATH=$PATH:$ACTIVATOR_HOME/bin
```

- Set apache as the forefront for play by adding virtualhost setting in apache conf file. Add below lines in conf file

```
<VirtualHost *:80>
  ProxyPreserveHost On
  ProxyPass  /excluded !
  ProxyPass / http://127.0.0.1:9000/
  ProxyPassReverse / http://127.0.0.1:9000/
</VirtualHost>
```

- Enable proxy mode by command `a2enmod proxy` and restart apache service.

B. Students Survey Result

We surveyed the instructors who are teaching introductory computer science courses and students from CMPE180-92 and CS149 to find the effectiveness of using LTIHub in their learning. Below is a result of students' survey. 65 students out of 105 from CMPE180-92 and 22 students out of 70 from CS149 participated in the survey. The survey result is used to compare students' perception of the amount of time spent for practice exercise vs the time spent that we got based on the timestamp of submission of a particular problem.

B.1 CMPE180-92 Student Responses

1. On average, how many times did you need to submit each problem until you received a perfect score?

○ 1 - 5 times	72.3 %
○ 5 - 10 times	20.0 %
○ 10 - 15 times	4.6 %
○ 15 - 20 times	3.1 %
○ More than 20 times	0.0 %

2. Please rate your agreement with this statement: These practice exercises are effective for reviewing course materials and preparing for the homework.

○ Strongly agree	69.2 %
○ Somewhat agree	29.2 %
○ Somewhat disagree	1.5 %
○ Strongly disagree	0.0 %

3. Please rate your agreement with this statement: These practice exercises are effective for reviewing course material and preparing for the exams.

○ Strongly agree	56.9 %
○ Somewhat agree	41.5 %
○ Somewhat disagree	1.5 %
○ Strongly disagree	0.0 %

4. Please rate your agreement with this statement: These practice exercises made me more confident with a new programming language.

○ Strongly agree	64.6 %
○ Somewhat agree	35.4 %
○ Somewhat disagree	0.0 %
○ Strongly disagree	0.0 %

5. Please rate your agreement with this statement: I would recommend these practice exercises to other students.

○ Strongly agree	75.4 %
○ Somewhat agree	20.0 %
○ Somewhat disagree	4.6 %
○ Strongly disagree	0.0 %

6. If you are asked to do more practice exercises without any credit in this course, would you be interested in doing it?

○ Very interested	46.2 %
○ Somewhat interested	38.5 %
○ Somewhat disinterested	7.7 %
○ Not interested	7.7 %

7. Would you be interested in having this kind of assignments in other programming courses?

○ Very interested	72.3 %
○ Somewhat interested	23.1 %

○ Somewhat disinterested	3.1 %
○ Not interested	1.5 %

8. On average, how many hours did you spend per week with these practice problems?

○ 0 hours	0.0 %
○ 1 to 3 hours	38.5 %
○ 4 to 6 hours	33.8 %
○ More than 6 hours	27.7 %

9. For effective practice, how many practice problems should be assigned in a typical week?

○ None	0.0 %
○ 1 to 3 problems	27.7 %
○ 4 to 6 problems	67.7 %
○ More than 6 problems	4.6 %

10. What grade are you expecting in this course?

○ Credit	78.5 %
○ No credit	21.5 %

11. How do you rate your prior programming experience before joining this course?

○ Very experienced	1.5 %
○ Intermediate experience	29.2 %
○ Basic familiarity	49.2 %
○ No experience	20.0 %

B.2 CS149 Student Responses

1. On average, how many times did you need to submit each problem until you received a perfect score?

○ 1 - 5 times	69.2 %
○ 5 - 10 times	19.2%
○ 10 - 15 times	7.7 %
○ 15 - 20 times	3.8 %
○ More than 20 times	0.0 %

2. Please rate your agreement with this statement: These practice exercises are effective for reviewing course materials and preparing for the homework.

○ Strongly agree	29.2 %
○ Somewhat agree	62.5 %
○ Somewhat disagree	8.3 %
○ Strongly disagree	0.0 %

3. Please rate your agreement with this statement: These practice exercises are effective for reviewing course material and preparing for the exams.

○ Strongly agree	9.1 %
○ Somewhat agree	77.3 %
○ Somewhat disagree	13.6 %
○ Strongly disagree	0.0 %

4. Please rate your agreement with this statement: These practice exercises made me more confident with a new programming language.

○ Strongly agree	36.4 %
○ Somewhat agree	50.0 %

○ Somewhat disagree	13.6 %
○ Strongly disagree	0.0 %

5. Please rate your agreement with this statement: I would recommend these practice exercises to other students.

○ Strongly agree	45.5 %
○ Somewhat agree	54.5 %
○ Somewhat disagree	0.0 %
○ Strongly disagree	0.0 %

6. If you are asked to do more practice exercises without any credit in this course, would you be interested in doing it?

○ Very interested	31.8 %
○ Somewhat interested	54.5 %
○ Somewhat disinterested	13.6 %
○ Not interested	0.0 %

7. Would you be interested in having this kind of assignments in other programming courses?

○ Very interested	40.9 %
○ Somewhat interested	54.5 %
○ Somewhat disinterested	4.5 %
○ Not interested	0.0 %

8. On average, how many hours did you spend per week with these practice problems?

○ 0 hours	9.1 %
○ 1 to 3 hours	68.2 %
○ 4 to 6 hours	22.7 %
○ More than 6 hours	0.0 %

9. For effective practice, how many practice problems should be assigned in a typical week?

○ None	0.0 %
○ 1 to 3 problems	72.7 %
○ 4 to 6 problems	22.7 %
○ More than 6 problems	4.6 %

10. What grade are you expecting in this course?

○ A	36.4 %
○ B	50.0 %
○ C	13.6 %
○ D/F	0.0 %

11. How do you rate your prior programming experience before joining this course?

○ Very experienced	13.6 %
○ Intermediate experience	54.5 %
○ Basic familiarity	27.3 %
○ No experience	4.5 %

C. Problems Offered as Practice Exercises in CMPE180-92

```
=========================
w1-1
/*
  Read a sequence of integers from cin that is terminated by a zero
  (which is not a part of the sequence). Print out the average of the
  first and last value of the sequence. If there is only one value, print
  it. If there are none, print 0. Do not use an array or vector.
*/
#include <iostream>
using namespace std;

int main()
{
   cout << "Average: " << avg << endl;
   return 0;
}
=========================
w1-2
/*
  Read a sequence of integers from cin that is terminated by a zero
  (which is not a part of the sequence). Print out all peaks, with
  one output per line containing just the peak, no text. A peak is a
  value that is strictly larger than the values that come before or
  after. The initial value is a peak if it is strictly larger than the
  second. The last value is a peak if it is strictly larger than its
  predecessor. If the sequence has length 1, the sole value is a peak.
  You may assume that the sequence is not empty.
  Do not use an array or vector.
*/
#include <iostream>
using namespace std;

int main()
{
}
=========================
w1-3
/*
  Read a sequence of strings from cin that is terminated by a "."
  (which is not a part of the sequence). Print out all strings
  and their lengths in a table, with a "|" separating the two table
  columns. The first column is left-justified with width 8,
  and the second column is right-justified with width 4. If a value
  doesn't fit the column, don't truncate it but overflow the column
  instead, without using any whitespace.
*/
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

int main()
{
   return 0;
}
=========================
w1-4
/*
  Read a sentence; that is, a sequence of strings from cin that is
  terminated by a string ending in "." (which is a part of the sequence).
  Print out the average length of all strings, reported to two digits
  and enclosed in [], in a sentence: The average length is [x.yy] characters.
*/
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
int main()
{
   return 0;
}
=========================
w1-5
/*
  Read a file and print out the average number of words per sentence.
  A sentence is a sequence of strings that is terminated by a string
  ending in "." (which is a part of the sequence). The file is terminated
  by a string "." which is not a sentence.Print out the average length of
  all sentences. You may assume there is at least one sentence.
*/
#include <iostream>
#include <fstream>
#include <string>

using namespace std;
int main()
{
   double average;
   string filename;
   cout << "File name: " << endl;
   cin >> filename;
   cout << "Average: " << average << endl;
   return 0;
}
=========================
w2-1
/**
   Computes the smallest of four values.
*/
int min(int a, int b, int c, int d)
{
   int result = a;
   if (b < result) result = b;
   if (c < result) result = c;
   if (d < result) result = d;
   return result;
}

/**
   Computes the average of the middle values of four given values
   (that is, without the largest and smallest value). Hint: Use the given
   min function. You may also want to define a max helper function or
   Take advantage of the fact that max can be computed from the min of
   the negative values.
*/
double middle(int a, int b, int c, int d)
{
   ...
}
=========================
w2-2
#include <string>
using namespace std;
/**
   Turn a string of the form firstname lastname or
   firstname middlename(s) lastname into lastname, firstname
   (and middlenames if present).
   For example, lastfirst("John Pierpont Flathead") should return
   "Flathead, John Pierpont". If the string contains no spaces, return
   it unchanged.
   Hint: If s is a string, then s.substr(i, n) is the substring
   starting at index i of length n. And if s and t are two strings,
   then s + t is the concatenation of the two strings. You will want
   to return a string of the form
   s.substr(i1, n1) + ", " + s.substr(i2, n2)
   except if s contains no spaces.
*/
string lastfirst(string s)
{
   ...
}
=========================
w2-3
#include "grades.h"
/**
   Given a letter grade (A, B, C, D, F) and a suffix (None, Plus, Minus),
   produce the numerical value of the grade. An A is a 4, B a 3,
   C a 2, D a 1, and F a zero. The suffix Plus adds 0.3, the suffix
   Minus subtracts 0.3. However, an A+ has value 4 and an F+ and F-
   have value zero.
 */
double gradeValue(Grade g, Suffix s)
{
   ...
}
=========================
w2-4
#include <string>
using namespace std;

/**
   Returns true if c is a vowel.
*/
bool isVowel(char c)
{
   return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u'
      || c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U';
}

/**
   Sets first to the index of the first vowel in the string s
   and last to the index of the last vowel in s. If s has no vowels,
   first and last are set to -1.
*/
void firstLastVowel(string s, int& first, int& last)
{
}
```

```
=========================
w2-5
#include <string>
using namespace std;

/**
   Returns true if c is a vowel.
*/
bool isVowel(char c)
{
   return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u'
      || c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U';
}

/**
   Swaps the first and last vowel in the string s.
   For example, if s is "Farewell", it is changed to "Ferewall".
*/
void swapFirstLastVowel(string& s)
{
   ...
}
=========================
w3-1
/**
   Computes the average of all positive elements in the given array.
   @param a an array of integers
   @param alen the number of elements in a
   @return the average of all positive elements in a, or 0 if there are none.
*/
double avgpos(int a[], int alen)
{
   ...
}
=========================
w3-2
#include <vector>
using namespace std;

/**
   Swaps the second and second-to-last value of a.
   For example, if a is {3, 1, 4, 1, 5, 9, 2, 6}
   after calling this method it is {3, 2, 4, 1, 5, 9, 1, 6}.
   If the array has length < 2, do nothing.
   @param a a vector of integers
```

```
*/
void swap2(vector<int>& a)
{
   ...
}
=========================
w3-3
#include <vector>
using namespace std;

/**
   Replaces each element in an array with the average of its
   neighbors. The first and last element are not changed. Do
   not declare any arrays or vectors.
   @param a an array
   @param n the length of the array
*/
void replavg(double a[], int n)
{
   ...
}

=========================
w3-4
The code below reads the following image from a file:
```

```
The pixels in the file are placed into a vector of vectors, one for each row
of the image. Each row is a vector of gray values between 0 and 255.
Color every seventh pixel black, starting from the top left corner..
```

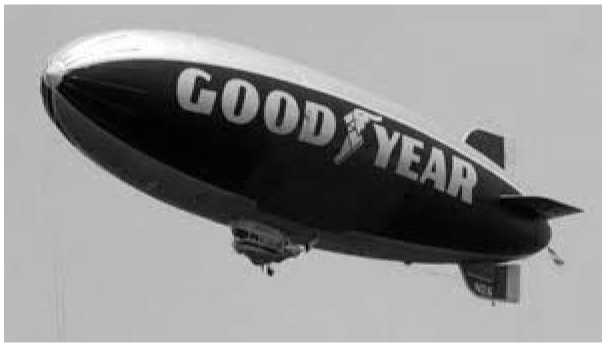
Complete the following file:

image10.cpp

==========================

W3-5

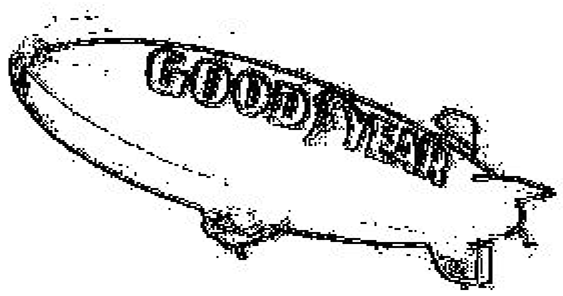
The code below reads the following image from a file:



The pixels in the file are placed into a vector of vectors, one for each row of the image. Each row is a vector of gray values between 0 and 255.

Detect edges in the image using the following approach:

For each pixel, compute the average gray level of the eight surrounding pixels (fewer at the image boundaries). If that average differs by more than 10 from the pixel value, color the pixel black (0). Otherwise, color it white (255). You need to assemble the result in a separate array since otherwise you will mix in pixels from the original and result.



Complete the following file:

```
image8.cpp
=========================
w4-1
/**
   Compute the minimum and maximum value in a non-empty array.
   @param arr the array
   @param n the length of the array
   @param min a pointer to a variable holding the minimum
   @param max a pointer to a variable holding the minimum
*/
void minmax(int* arr, int n, int* min, int* max)
{
   ...
}
=========================
w4-2
#include <cstring>
using namespace std;

/**
   Given a '\0'-terminated character array, split it by replacing
   each space in the character array with a '\0'. Return the number
   of strings into which you have split the input.
*/
int split(char* words)
{
   ...
}
=========================
w4-3
/*
  The print method receives as input a string followed by an asterisk
  and an integer. Print out the string as many time as indicated
  by the integer. For example, when called as print("Hi*3"),
  you print HiHiHi. If no integer is specified, print the string
  once.
  Hint: Look for the '*' starting from the back of the string.
  Then call atoi, passing a pointer to the integer starting after
  the '*'. You can also replace the '*' with a '\0' for easy
  printing of the first part.
*/

#include <iostream>
#include <cstdlib>
```

```
#include <cstring>
using namespace std;

void print(char* arg)
{
   ...
}
=========================
w4-4

#include <iostream>
#include <cstring>
using namespace std;

/*
  Repeats a string n times and places it into a buffer, filling
  it up as much as possible, and providing a terminating 0.
  Use strncpy.
  @param str a string
  @param n an integer
  @param result a character array to hold the result
  @param sz the size of result
*/
void repeat(const char* str, int n, char result[], int sz)
{
   if (sz <= 0) return;
   int len = strlen(str);
   char* p = result;
   for (int i = 0; i < n && p < result + sz; i++)
   {
      strncpy(...)
      ...
   }
   ...
}
=========================
w4-5
#include <cstring>
/*
  Return a pointer to the nth occurrence of a repeated character
  in the given string. For example, if str is "occurrence" and n is 2,
  return a pointer to the first r.
*/
const char* nthrep(const char* str, int n)
{
   const char* p = str;
   const char* r = NULL;
   int len = strlen(str);
   int count = 0;
   while (p < str + len && count < n)
   {
      ...
   }
   return r;
}
=========================
w5-1
/*
  The First National Bank of Stroustrup provides accounts in which
  deposits and balance inquiries have no charge, but there is a
  $10 charge for withdrawals if after the withdrawal the balance
  falls below $1000. Implement the withdraw member function.
*/

#include "BankAccount.h"
BankAccount::BankAccount()
{
   balance = 0;
}

void BankAccount::deposit(double amount)
{
   double newBalance = balance + amount;
   balance = newBalance;
}

double BankAccount::getBalance()
{
   return balance;
}
=========================
w5-2
/*
  The Second National Bank of Stroustrup provides accounts in which
  deposits and balance inquiries have no charge. After a deposit,
  the first two withdrawals are free, but any further withdrawals
  have a charge of $10 until the next deposit. Implement the
  deposit and withdraw member functions.
*/
#include "BankAccount.h"
```

```
BankAccount::BankAccount()
{
   balance = 0;
   withdrawals = 0;
}

double BankAccount::getBalance()
{
   return balance;
}
=========================
w5-3
#include "Time.h"

/*
  Provide the hours and minutes member functions.
 */
Time::Time(int hr, int min)
{
   minutesSinceMidnight = 0;
   add_minutes(60 * hr + min);
}
...
void Time::add_minutes(int minutes)
{
   minutesSinceMidnight += minutes;
   minutesSinceMidnight %= 24 * 60;
   if (minutesSinceMidnight < 0) minutesSinceMidnight += 24 * 60;
}

/**
   A time of the day between 0:00 and 23:59
*/
class Time
{
 public:
   /**
       @param hr the hour (between 0 and 23)
       @param min the minutes (between 0 and 59)
       */
   Time(int hr, int min);
   /**
       @return the hours in military time (between 0 and 23).
   */
   int hours();
   /**
      @return the minutes (between 0 and 59)
   */
   int minutes();
   /**
      @param the minutes to add or subtract (if negative)
   */
   void add_minutes(int minutes);
 private:
   int minutesSinceMidnight;
};
========================
w5-4

#include <cmath>
#include "point.h"

Point::Point(double xvalue, double yvalue)
{
   x = xvalue;
   y = yvalue;
}

double Point::get_x() { return x; }
double Point::get_y() { return y; }

/**
   Compute the distance between this point and another point.
   @param other the other point
   @return the distance between them.
*/
double Point::distance(Point other)
{
   ...
}
========================
w5-5

#include <cmath>
#include "point.h"

Point::Point(double xvalue, double yvalue)
{
   x = xvalue;
   y = yvalue;
}
double Point::get_x() { return x; }
double Point::get_y() { return y; }

/**
   Compute the midpoint between this point and another point.
   @param other the other point
   @return the point halfway between them.
*/
Point Point::midpoint(Point other)
{
   ...
}
=========================
w6-1
#include "path.h"
/*
  Produce a class Path that represents a path of points.
  Use a vector to store Point objects.
*/

void Path::add(int x, int y)
{
   ...
}

void Path::add(const Point& p)
{
   ...
}

Point& Path::at(int index)
{
   ...
}

int Path::length()
{
   ...
}

void Path::print()
{
   for (int i = 0; i < length(); i++)
   {
      if (i > 0) cout << "->";
      cout << "(" << at(i).x() << "," << at(i).y() << ")";
   }
   cout << endl;
}
=========================
w6-2
#include "path.h"
/*
  Provide a function backtrack that, given a path, produces and prints
  a path that starts like the given path and then goes back to
  the beginning. For example, when given the path (1,2)->(3,4)->(5,6)
  you should produce and print (1,2)->(3,4)->(5,6)->(3,4)->(1,2).
  DO NOT COPY the Path parameter into the function. Use a reference
  parameter.
  DO NOT COPY Point objects. Note that Point::at(int) returns a
  reference, and Point::add(Point&) receives a reference.
  The Point class is instrumented to print messages for all copies.
*/
void backtrack(...)
{
   for (int i = p.length() - 2; i >= 0; i--)
   ...
   p.print();
}
=========================
w6-3
/**
   This Path class uses an array, not a vector, to store the points
   of the path. Provide the missing constructor and member function.
*/
#include "path.h"
Path::Path(int n)
{
   ...
}

/*
  Produce a class Path that represents a path of points.
  Use a vector to store Point objects.
*/
void Path::add(int x, int y)
{
   ...
```

```
}

Point& Path::at(int index)
{
   return _points[index];
}

int Path::length()
{
   return _length;
}

void Path::print()
{
   for (int i = 0; i < length(); i++)
   {
      if (i > 0) cout << "->";
      Point& p = at(i);
      cout << "(" << p.x() << "," << p.y() << ")";
   }
   cout << endl;
}
=========================
w6-4
/**
   The eagle-eyed among you will have noticed that in the preceding
   exercise many points were constructed, and few were destroyed.
   Fix that by providing a destructor forr the Path class. You will
   also need to provide the constructor and add method again since
   we don't want to give away the solution to the preceding problem.
*/
#include "path.h"
Path::Path(int n)
{
   ...
}

// Destructor...
/*
  Produce a class Path that represents a path of points.
  Use a vector to store Point objects.
*/
void Path::add(int x, int y)
{
   ...
```

```
}

Point& Path::at(int index)
{
   return _points[index];
}

int Path::length()
{
   return _length;
}

void Path::print()
{
   for (int i = 0; i < length(); i++)
   {
      if (i > 0) cout << "->";
      Point& p = at(i);
      cout << "(" << p.x() << "," << p.y() << ")";
   }
   cout << endl;
}
=========================
w6-5
/**
   Now we make a teeny-tiny change to the runner. Instead of

      Path p2(2);

   it is

      Path p2 = p1;

   The result: disaster. Now both p1 and p2 get destroyed, and
   the destructor deletes the same memory block twice. If you are
   lucky, your program dumps core. If you are unlucky, it does
   something completely random.

   Fix this by implementing a copy constructor. Allocate
   a new array of the same capacity as the original,
   and copy over all points

   You will also need to provide the constructor, add method, and
   destructor  again since we don't want to give away the solution
   to the preceding problems. Sorry about that.
```

```
*/
#include "path.h"

Path::Path(int n)
{
   ...
}

// Destructor...
// Copy constructor
Path::Path(const Path& other)
{
   _capacity = other._capacity;
   _length = other._length;
   ...
}

/*
  Produce a class Path that represents a path of points.
  Use a vector to store Point objects.
*/
void Path::add(int x, int y)
{
   ...
}

Point& Path::at(int index)
{
   return _points[index];
}

int Path::length()
{
   return _length;
}

void Path::print()
{
   for (int i = 0; i < length(); i++)
   {
      if (i > 0) cout << "->";
      Point& p = at(i);
      cout << "(" << p.x() << "," << p.y() << ")";
   }
   cout << endl;
```

```
}
=========================
w7-1
#include "list.h"
/*
   This function removes every second element from the given linked
   list. For example, if a list contains 12 15 26 25 11 (and then NULL),
   it is modified to contain 12 26 11 (and then NULL).
*/
void remove_every_second(NodePtr head)
{
   ...
}
=========================
w7-2
#include "list.h"
/*
   This function removes every even element from the given linked
   list. For example, if a list contains 16 26 25 12 11 (and then NULL),
   it is modified to contain 25 11 (and then NULL).
*/
void remove_every_even(NodePtr& head)
{
   ...
}
=========================
w7-3
/*
Your task is to break a positive number into its individual digits, for
example, to turn 1729
into 1, 7, 2, and 9. It is easy to get the last digit of a number n as n % 10.
But that gets
the numbers in reverse order. Solve this problem with a stack. Your program
should
ask the user for an integer, then print its digits separated by spaces.
*/

#include <iostream>
using namespace std;
#include "stack.h"

void print_digits(int n)
{
   Stack digit_stack;
   while (n > 0)
   {
   ...
   }
   while (...)
   {
       ...
       cout << ... << " ";
   }
   cout << endl;
}
=========================
w7-4
/*
  Use a stack to find out if a sequence of HTML tags is balanced.
  Your program will read in the tags, one at a time, such as
  <ol>
  <li>
  </li>
  <li>
  </li>
  </ol>
  If everything is ok, print "OK"
  When you find a closing tag that doesn't match its opening
  tag, print "BAD " followed by the tag name.
  If you reached the end of input and there are missing closing
  tags, print "UNCLOSED" followed by all unclosed tag names,
  starting with the last unclosed one.
*/

#include <iostream>
using namespace std;
#include "stack.h"

int main()
{
   string tag;
   Stack tag_stack;
   while (cin >> tag)
   {
       tag = tag.substr(1, tag.length() - 2); // Strip off "<...>"
       if (tag.substr(0, 1) == "/")
       {
       tag = tag.substr(1); // Strip off "/"
       ...
       }
   }
   if (...)
      cout << "OK" << endl;
   else
   {
      cout << "UNCLOSED ";
      while (...)
      cout << ... << " ";
      cout << endl;
   }

   return 0;
}
=========================
w7-5
/*
A queue is often useful when you need to break a task into simpler tasks. Here
you will use a queue to enumerate all permutations of a string.

Suppose you want to find all permutations of the string meat.

Add the string +meat on the queue.
While the queue is not empty
   Remove a string from the queue
   If that string ends in a + (such as tame+)
      Remove the + and print the string
   Else
      Remove each letter in turn from the right of the +.
      Insert it just before the +.
      Add the resulting string on the stack.
      For example, after removing e+mta, you add em+ta, et+ma, and ea+mt.
*/

#include <iostream>
using namespace std;
#include "queue.h"

int main()
{
   string word;
   cin >> word;
   Queue work_queue;
   work_queue.add("+" + word);
   while (...)
   {
      string str = ...
      int i = 0;
      while (str.substr(i, 1) != "+") i++;
      if (i == str.length() - 1)
      cout << str.substr(0, i) << endl;
      else
      {
      for (int j = i + 1; j < str.length(); j++)
      {
             // Make a string consisting of
             // the part of before the +
             // the letter at index j
             // +
             // everything after the + and before index j
             // everything after index j
             string to_add = str.substr(...) + str.substr(j, 1) + "+" +
str.substr(i + 1, j - i - 1) + str.substr(...);
             ...
      }
      }
   }
   return 0;
}
=========================
w9-1
#include <list>
using namespace std;
/**
   Remove the elements at position 0, 2, 4, 6, ..., of the
   linked list.
   Use an iterator.
*/
void removeEvenPositions(list<int>& lst)
{
   ...
}
=========================
w9-2
#include <list>
using namespace std;
/**
   Remove the even elements of the linked list.
   Use an iterator.
*/
void removeEven(list<int>& lst)
```

```
{
   ...
}
========================
w9-3
#include <list>
using namespace std;
/**
   Swap neighboring elements of this linked list. If the
   length is odd, leave the last element unchanged.
*/
void swapNeighbors(list<int>& lst)
{
   ...
}
========================
w9-4
#include <list>
using namespace std;

/**
   Swap the first and second half of the list. For example,
   if lst contains 1 2 4 8 16 32, afterwards it should contain
   8 16 32 1 2 4.  If the list has odd length, the middle
   element should become the last element of the result.

   Use iterators.
*/
void swapHalves(list<int>& lst)
{
   ...
}
========================
w9-5
#include <list>
using namespace std;

/**
      Changes the given lists so that they "cross over" like this:

      a: 0 ... n-1 n ...
                   x
      b: 0 ... m-1 m ...

      For example, if a is [1, 2, 3, 4], b is [5, 6, 7],
      n is 2 and m is 1, then a becomes [1, 2, 6, 7] and
      b becomes [5, 3, 4].

      Use iterators.

      You may assume that n < a.size() and m < b.size().
   */
void crossOver(list<int>& a, int n, list<int>& b, int m)
{
   ...
}
=========================
w10-1
#include <string>
using namespace std;

/**
   Return true if the given string contains at least two
   different characters. Use recursion. Do not use loops.
   Hint: If str[0] is not the same as str[1], you have
   your answer. Otherwise, look at str.substr(1) (the substring of str
   from position 1 to the end).
*/
bool someDifferent(string str)
{
   ...
}
=========================
w10-2
#include <string>
using namespace std;

/**
   Mix two strings by alternating characters from them. If one string
   runs out before the other, just pick from the longer one.
   For example, mix("Fred", "Wilma") is "FWrieldma".
   Use recursion. Do not use loops.
   Hint: str.substr(1) is the substring of str from position 1 to
   the end.
*/
string mix(string a, string b)
{
   ...
}
=========================
w10-3
#include <string>
using namespace std;

/**
   Given a string, return a string of all lowercase letters contained in it.
   Use recursion. Do not use loops.
   A lowercase letter is a character between 'a' and 'z' inclusive.
   Hint: str.substr(1) is the substring of str from position 1 to
   the end.
*/
string lcl(string str)
{
   ...
}
=========================
w10-4
#include <string>
#include <vector>
using namespace std;

void lclHelper(string str, string currentGroup, vector<string>& lclGroups)
{
   ...
}

/**
   Given a string, return a vector of all substrings consisting of
   lowercase letters contained in it. For example, if str is "I like 7 and
13",
   return a vector containing "like", and "and".
   Use a recursive helper method. Do not use loops.
   A lowercase letter is a character between 'a' and 'z' inclusive.

*/
vector<string> lclGroups(string str)
{
   vector<string> result;
   lclHelper(str, "", result);
   return result;
}
=========================
w10-5
#include <string>
#include <vector>
```

```
using namespace std;
...

/**
   Given a vector of floating-point numbers, return the product quotient
   v[0] / v[1] * v[2] / v[3] * ...
   If the vector is empty, the product quotient is 1.
   Use a recursive helper method. Do not use loops.

*/
double prodQuot(vector<double> v)
{
   return prodQuotHelper(v, 0);
}
=========================
w10-6
#include <string>
#include "list.h"
using namespace std;

/**
   Look at the recursively defined List class in list.h.
   A List is either empty or it has a head and a tail.
   Implement the show function that yields a visual
   string of the list elements, separated by ->, and
   surrounded by [].
   If the list is empty, return "[]".
*/
string show(const List& lst)
{
   ...
}
=========================
w11-1
#include "gradebook.h"

/**
   Implement the member functions of the GradeBook class that uses
   a map to keep student scores.
*/

void GradeBook::add_student(string student)
{
   . . .
}
```

```
void GradeBook::add_score(string student, int score)
{
   . . .
}

int GradeBook::get_score(string student)
{
   . . .
}
=========================
w11-2
#include "gradebook.h"

/**
   The gradebook of the preceding problem wasn't very useful
   because it only keeps a single score for each student.

   In this problem, we make the gradebook more realistic by
   keeping a map of tasks to scores for each student.

*/

void GradeBook::add_student(string student)
{
   . . .
}

void GradeBook::set_score(string student, string task, int score)
{
   . . .
}

int GradeBook::get_score(string student, string task)
{
   . . .
}
=========================
w11-3
#include "gradebook.h"

/**
   Now we change the gradebook again so that one can increment the
   score of a task (like in the first program).
*/
```

```
void GradeBook::add_student(string student)
{
   . . .
}

void GradeBook::add_score(string student, string task, int score)
{
   . . .
}

int GradeBook::get_score(string student, string task)
{
   . . .
}
=========================
w11-4
#include <vector>
#include <map>
using namespace std;

/**
   Makes a map associating integers with the number of words
   whose length is the given integer.
   @param words a vector of strings
   @return the map

*/
map<int, int> lengthFrequency(vector<string> words)
{
   ...
}
=========================
w11-5
#include <vector>
#include <map>
#include <set>
using namespace std;

/**
  Makes a map associating integers with the words
  having that length.
  @param words an array of strings
  @return the map
*/
```

```
map<int, set<string>> wordsByLength(vector<string> words)
{
   ...
}
=========================
w11-6
#include <map>
#include <set>
using namespace std;

/**
   Invert the given map: Given an int->int map, return a map
   from int->set<int> whose keys are the values of the original
   map, and whose values are the keys from the original map mapping
   to the given values. For example, the inverse of the map

   1 -> 2, 2 -> 2, 3 -> 3

   is the map

   2 -> { 1, 2 }
   3 -> { 3 }
*/

map<int, set<int>> invert(const map<int, int>& m)
{
   ...
}
```