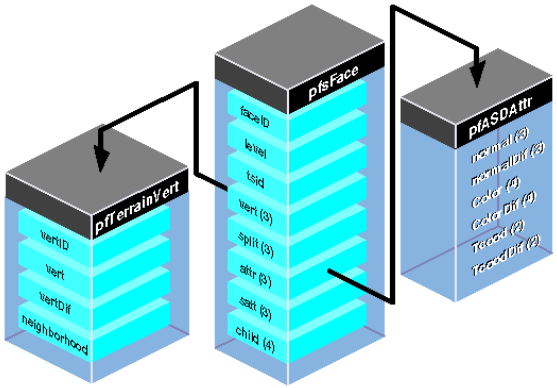
Struktur Data – Pertemuan 13

List Rekursif



P r a j a n t o W a h y u A d i

prajanto@dsn.dinus.ac.id

+6285 641 73 00 22

Rencana Kegiatan Perkuliahan Semester

#	Pokok Bahasan
~~1~~	~~Pengenalan Struktur Data~~
~~2~~	~~ADT Stack & Queue~~
~~3~~	~~List Linear~~
~~4~~	~~List Linear~~
~~5~~	~~List Linear~~
~~6~~	~~Representasi Fisik List Linear~~
~~7~~	~~Variasi List Linear~~
8	**Ujian Tengah Semester**

#	Pokok Bahasan
~~9~~	~~Variasi List Linear~~
~~10~~	~~Variasi List Linear~~
~~11~~	~~Stack dengan Representasi List~~
~~12~~	~~Queue dengan Representasi List~~
13	**List Rekursif**
14	Pohon dan Pohon Biner
15	Studi Kasus Multi List
16	**Ujian Akhir Semester**

Konten

1. Pendekatan Iteratif dan Rekursif
2. Membalik List secara Iteratif
3. Fungsi Rekursif pada Memory
4. Membalik List secara Rekursif

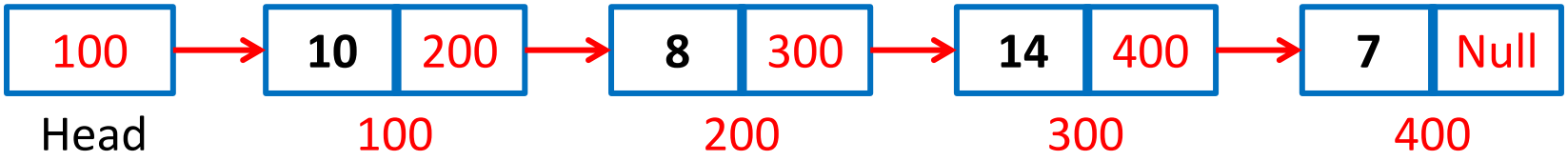
Pendekatan Iteratif dan Rekursif

- Pendekatan iteratif menggunakan proses perulangan (loop) untuk menyelesaikan masalah
- Dalam konteks prosedural kita memiliki “loop” sebagai mekanisme untuk mengulang

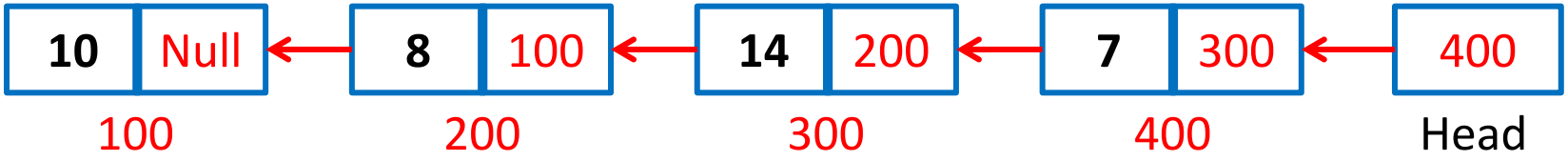
- Suatu entitas disebut rekursif jika pada definisinya terkandung dirinya sendiri
- Program prosedural juga dapat bersifat rekursif
- Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri

Studi Kasus: Membalik List secara Iteratif

- Input:

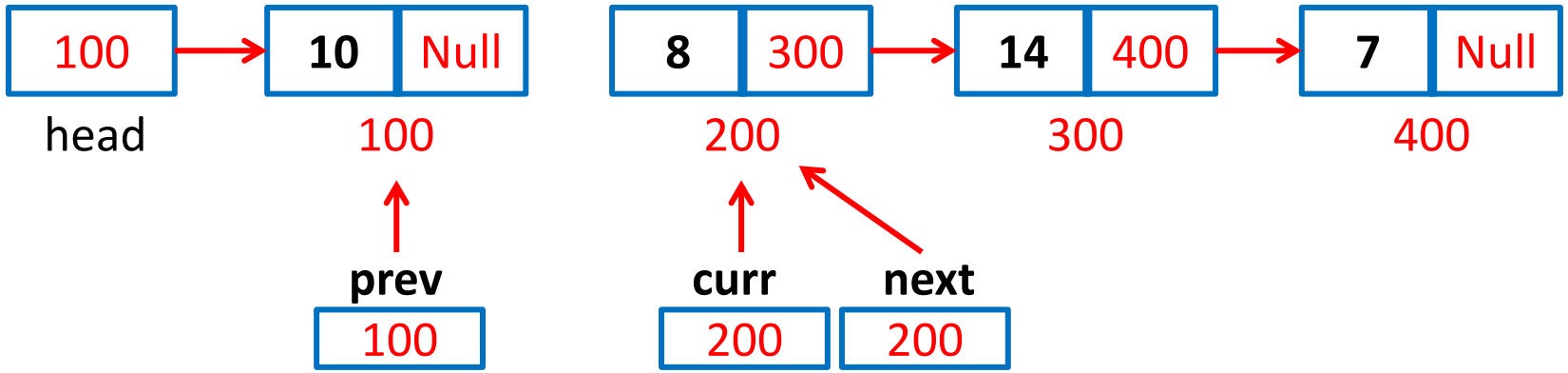


- Output:



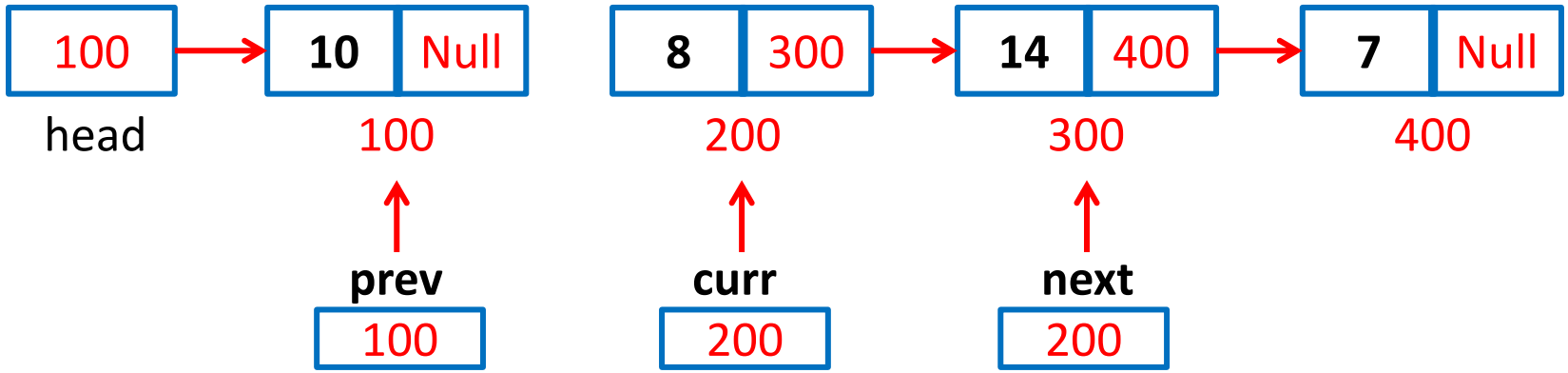
Studi Kasus: Membalik List secara Iteratif

- Diperlukan tiga buah variabel pointer:
  - current
    membentuk link baru sekaligus memutus link sebelumnya
  - next
    memindahkan pointer current ke node selanjutnya setelah link sebelumnya diputus
  - previous
    menyimpan alamat node sebelumnya setelah link diputus



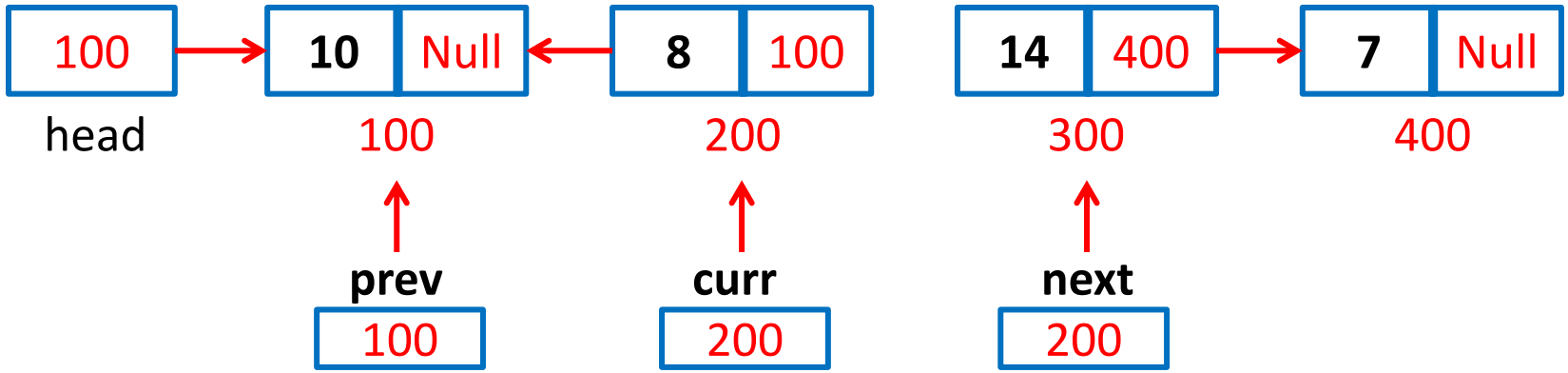
Studi Kasus: Membalik List secara Iteratif

- Diperlukan tiga buah variabel pointer:
  - current
    membentuk link baru sekaligus memutus link sebelumnya
  - next
    memindahkan pointer current ke node selanjutnya setelah link sebelumnya diputus
  - previous
    menyimpan alamat node sebelumnya setelah link diputus



Studi Kasus: Membalik List secara Iteratif

- Diperlukan tiga buah variabel pointer:
  - current
    membentuk link baru sekaligus memutus link sebelumnya
  - next
    memindahkan pointer current ke node selanjutnya setelah link sebelumnya diputus
  - previous
    menyimpan alamat node sebelumnya setelah link diputus



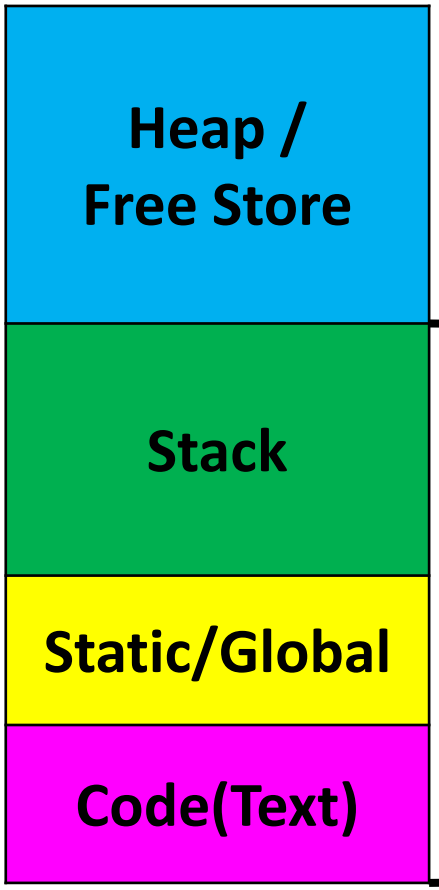
Studi Kasus: Membalik List secara Iteratif

//Fungsi membalik list secara Iteratif

```
void balikListIteratif(){
    Node *curr, *prev, *next;
    curr = head;
    prev = NULL;
    while(curr != NULL){
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    head = prev;
}
```

Fungsi Rekursif pada Memory

- Memori yang dialokasikan pada sebuah program/ aplikasi umumnya dibagi menjadi 4 bagian:

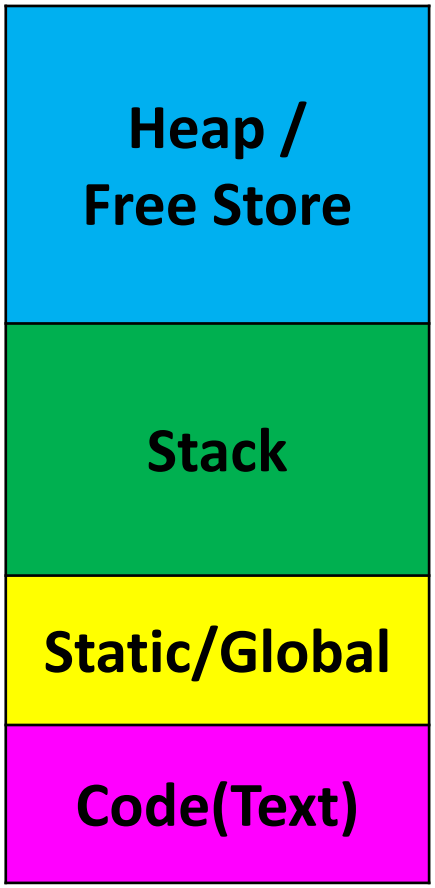


Bersifat **Dinamis**:
Ukuran memori dapat berubah ketika program dijalankan

Bersifat **Statis**:
- ukuran memori ditentukan ketika kompilasi
- ukuran memori tidak dapat berubah

Fungsi Rekursif pada Memory

- Memori yang dialokasikan pada sebuah program/ aplikasi umumnya dibagi menjadi 4 bagian:



Memori yang dapat di-*request* selama program berjalan

Menyimpan semua informasi tentang pemanggilan **fungsi** untuk menyimpan semua variabel lokal

Menyimpan variabel global selama program berjalan

Menyimpan semua perintah yang akan dijalankan/dieksekusi

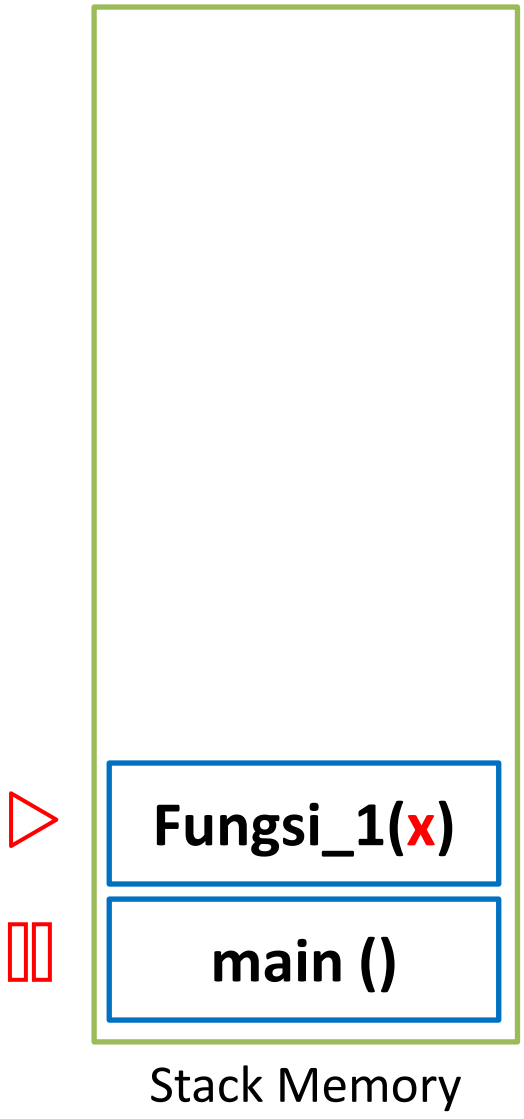
Fungsi Rekursif pada Memory

- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu

Fungsi Rekursif pada Memory

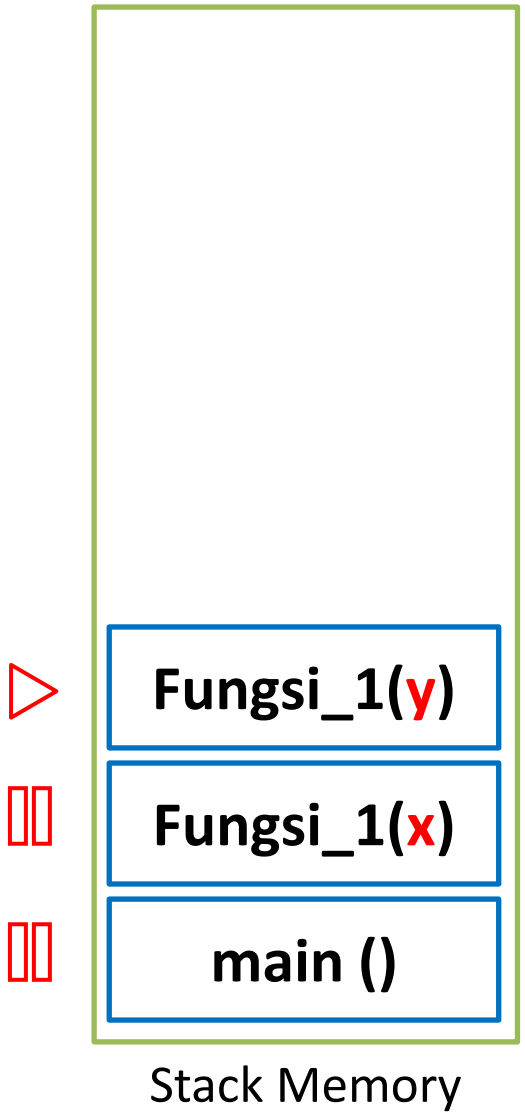
- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan ‘ditumpuk’ (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*

Fungsi Rekursif pada Memory



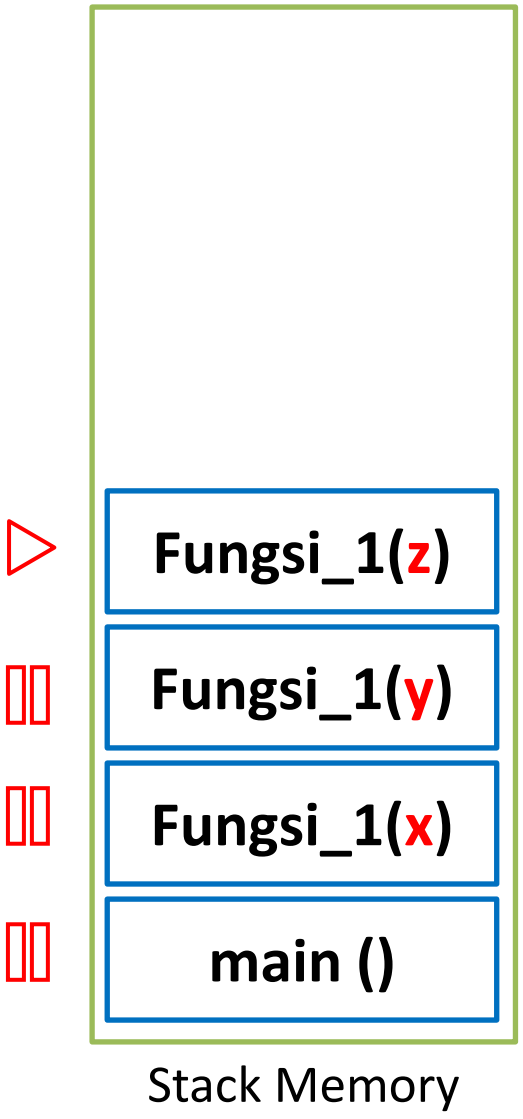
- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan ‘ditumpuk’ (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*

Fungsi Rekursif pada Memory



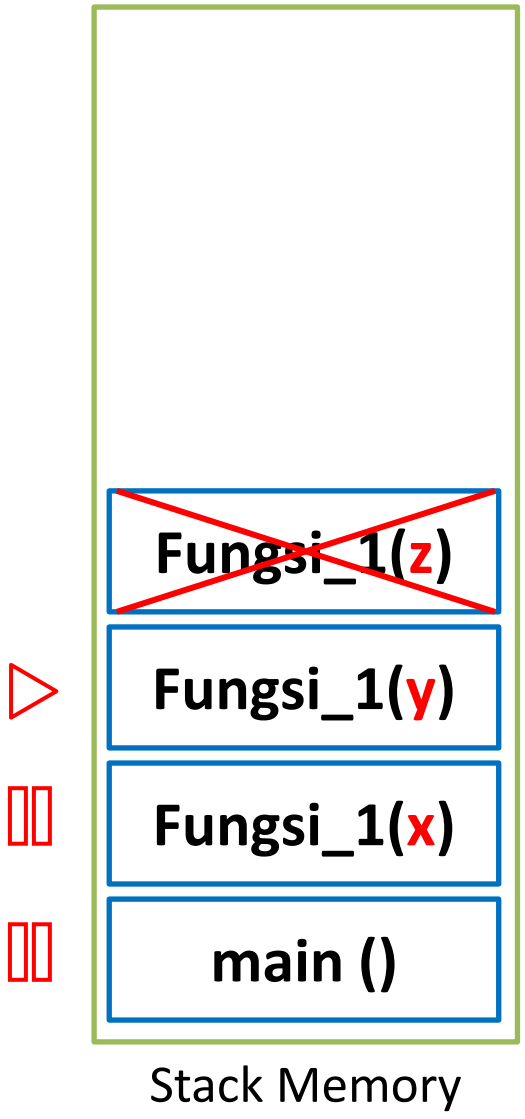
- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan 'ditumpuk' (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*

Fungsi Rekursif pada Memory



- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan ‘ditumpuk’ (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*

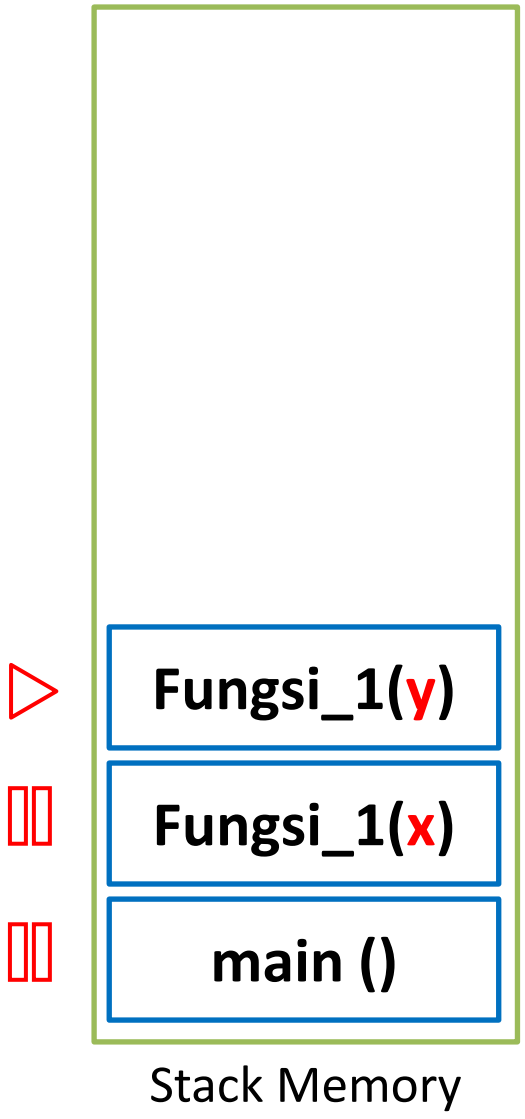
Fungsi Rekursif pada Memory



Stack Memory

- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan 'ditumpuk' (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*
- Jika fungsi pada tumpukkan paling atas selesai dijalankan, maka fungsi yang berada pd tumpukkan dibawahnya akan dijalankan kembali (resume).

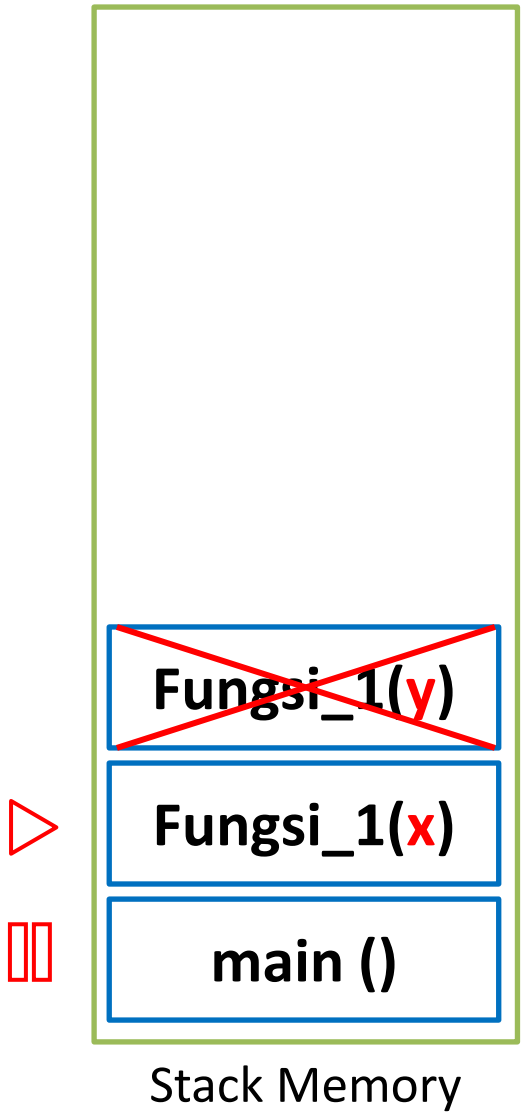
Fungsi Rekursif pada Memory



Stack Memory

- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan 'ditumpuk' (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*
- Jika fungsi pada tumpukkan paling atas selesai dijalankan, maka fungsi yang berada pd tumpukkan dibawahnya akan dijalankan kembali (resume).

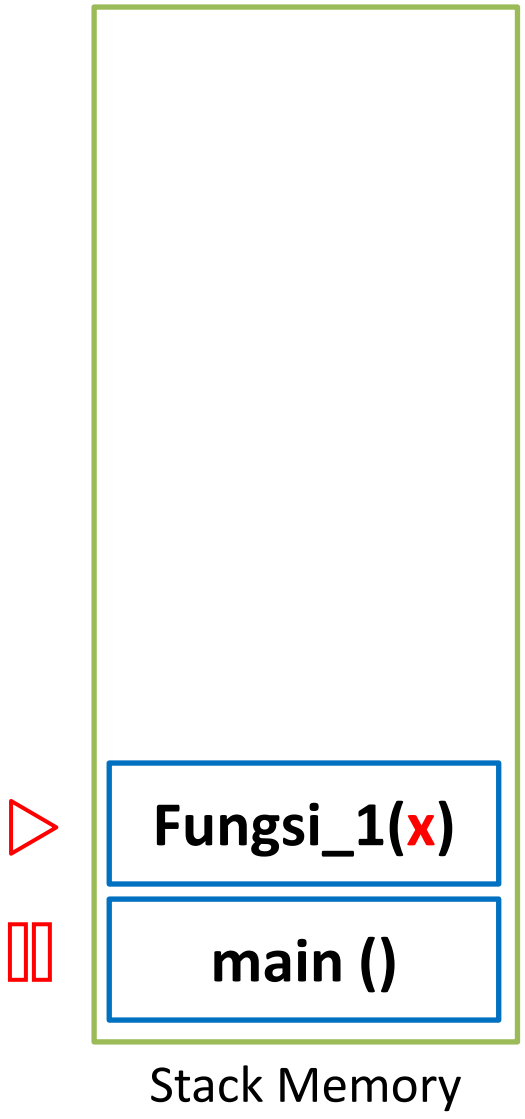
Fungsi Rekursif pada Memory



Stack Memory

- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan 'ditumpuk' (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*
- Jika fungsi pada tumpukkan paling atas selesai dijalankan, maka fungsi yang berada pd tumpukkan dibawahnya akan dijalankan kembali (resume).

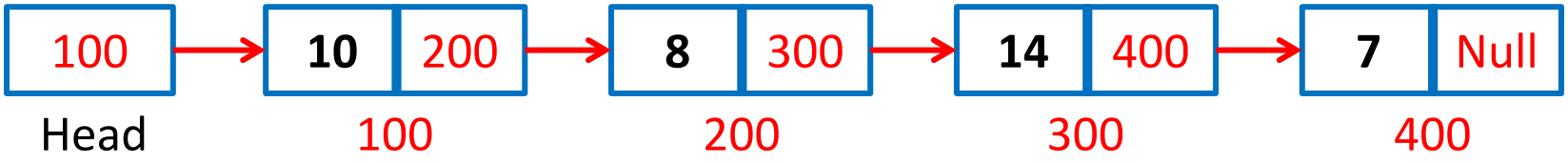
Fungsi Rekursif pada Memory



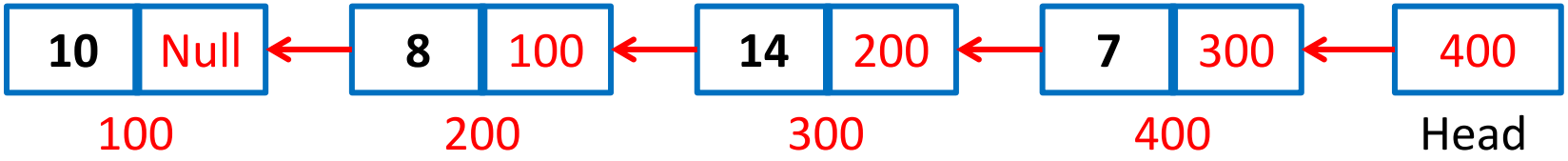
- *Stack memory* dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada *stack memory*
- Fungsi baru akan 'ditumpuk' (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada *stack memory*
- Jika fungsi pada tumpukkan paling atas selesai dijalankan, maka fungsi yang berada pd tumpukkan dibawahnya akan dijalankan kembali (resume).

Studi Kasus: Membalik List secara Rekursif

- Input:



- Output:

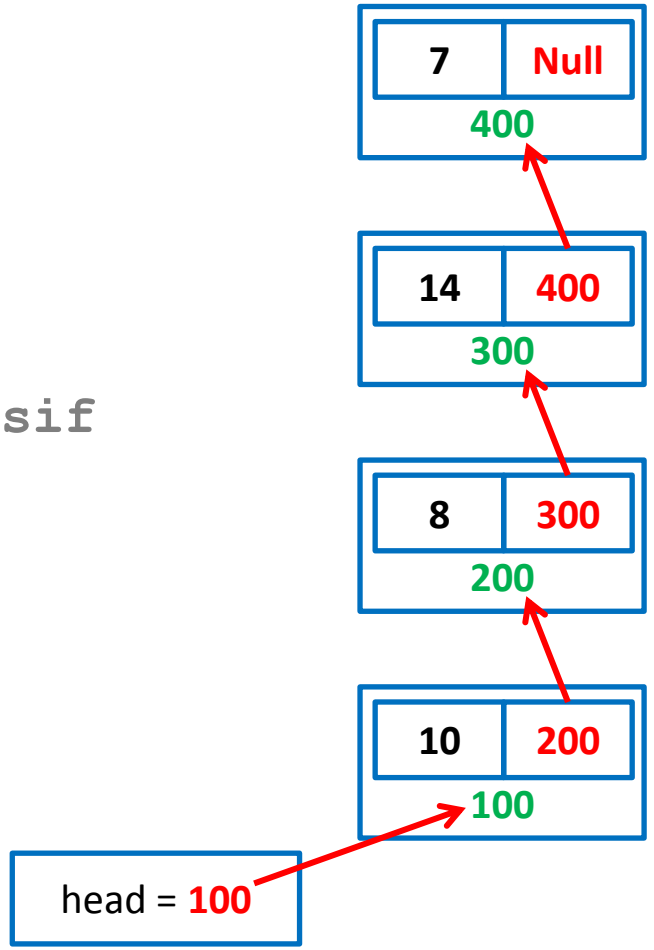


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next); //rekursif
    p->next->next = p;
    p->next = NULL;
}
```

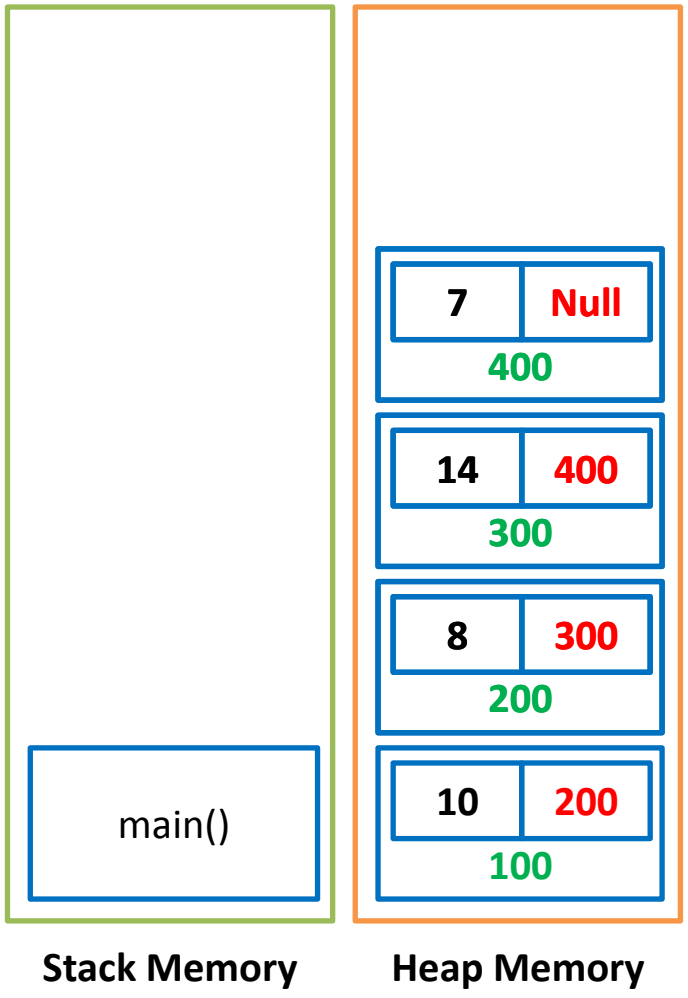


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

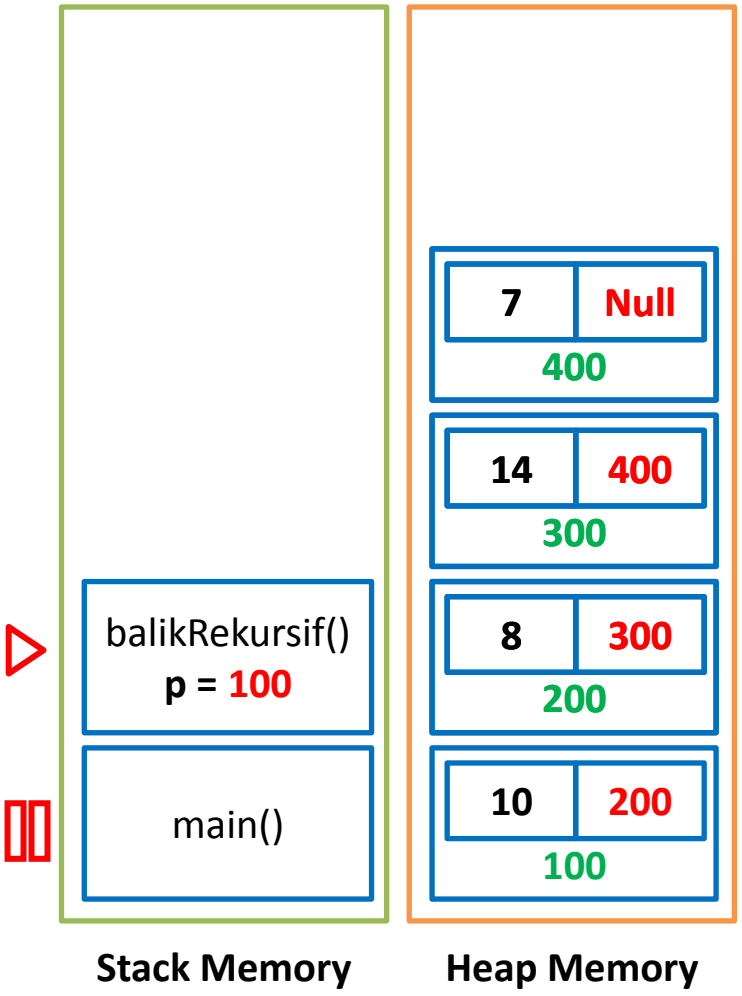


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=100
}
```

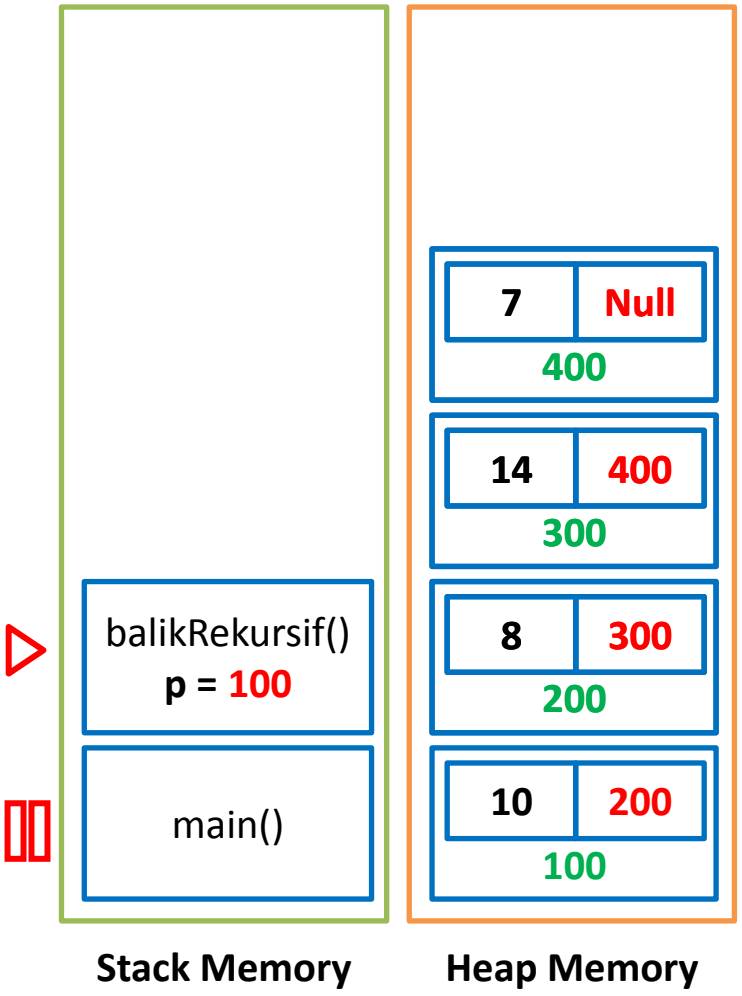


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

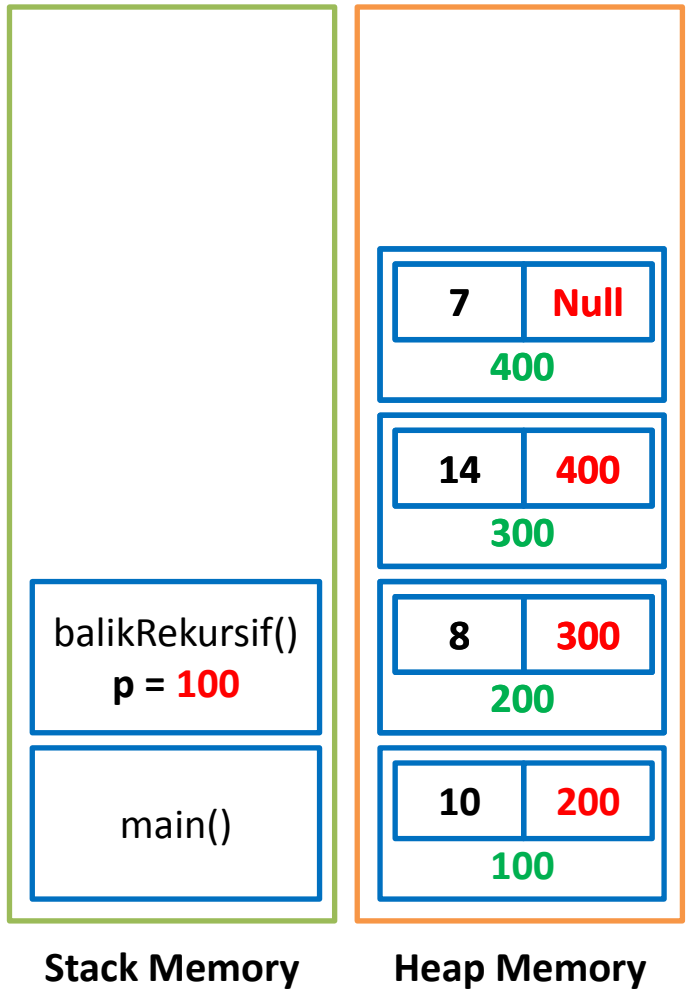


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

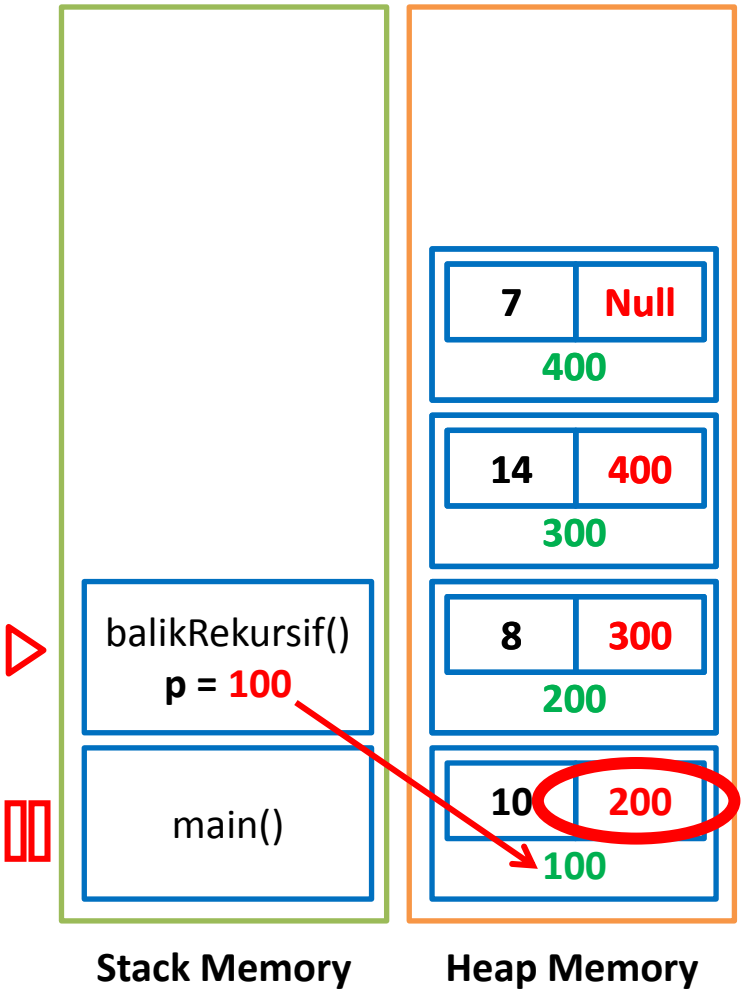


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

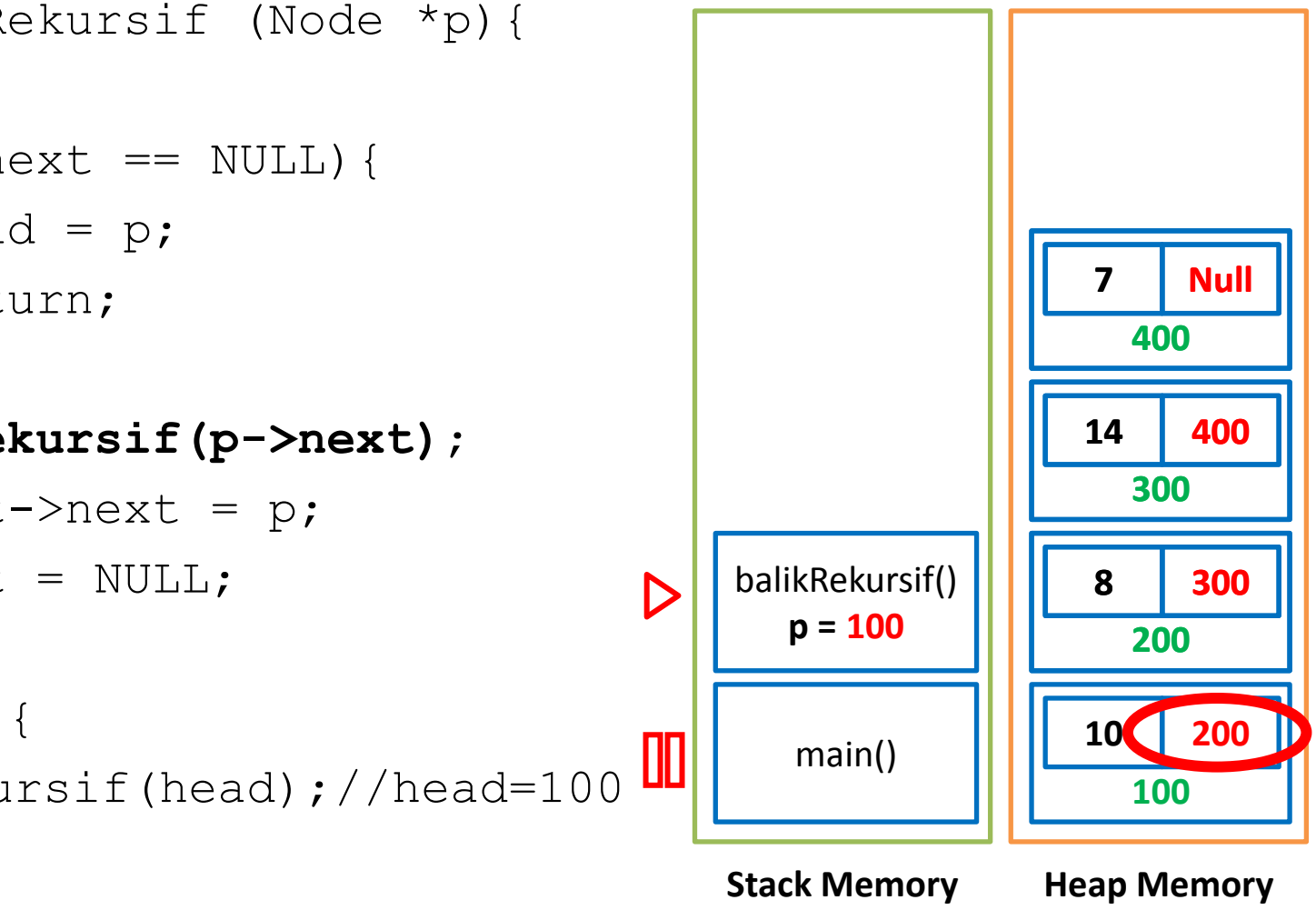


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

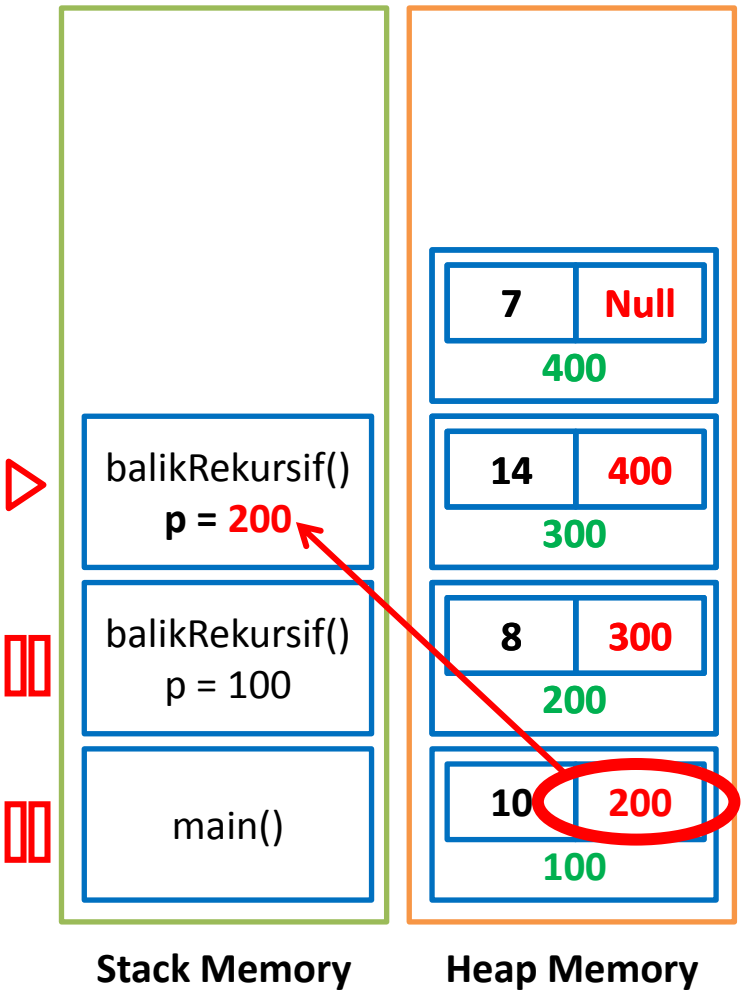


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

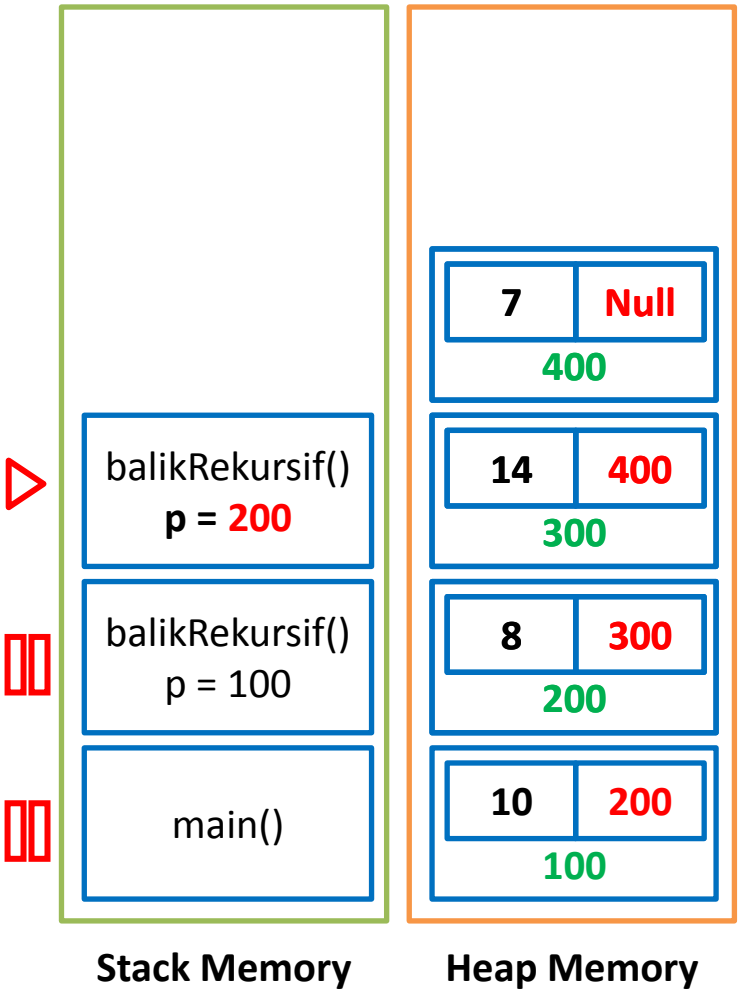


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

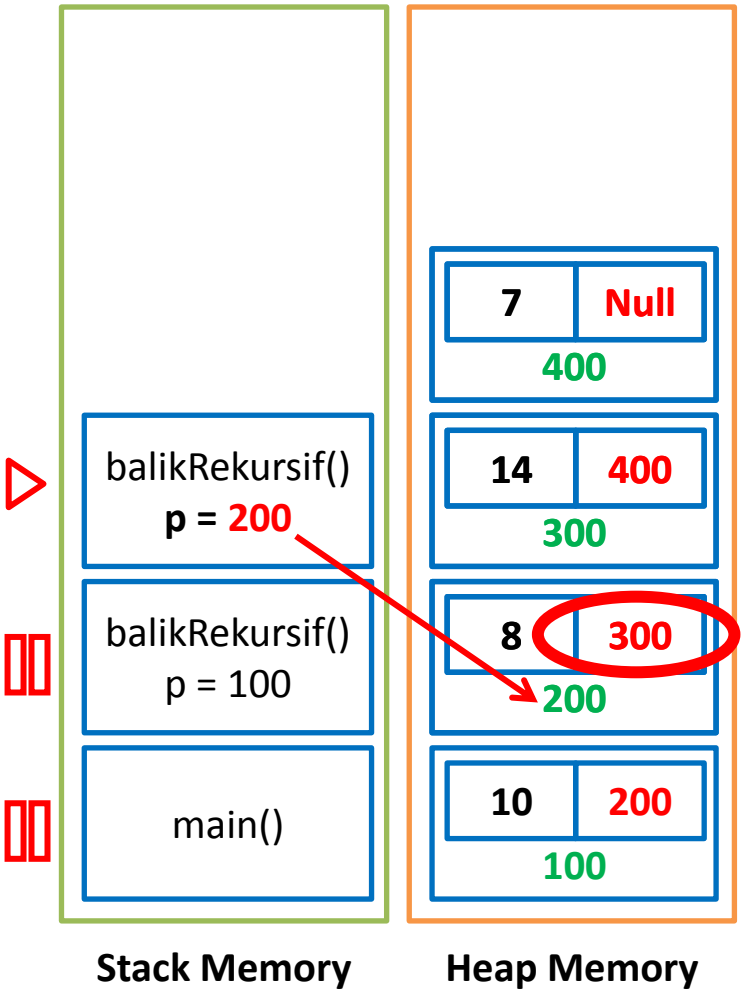


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

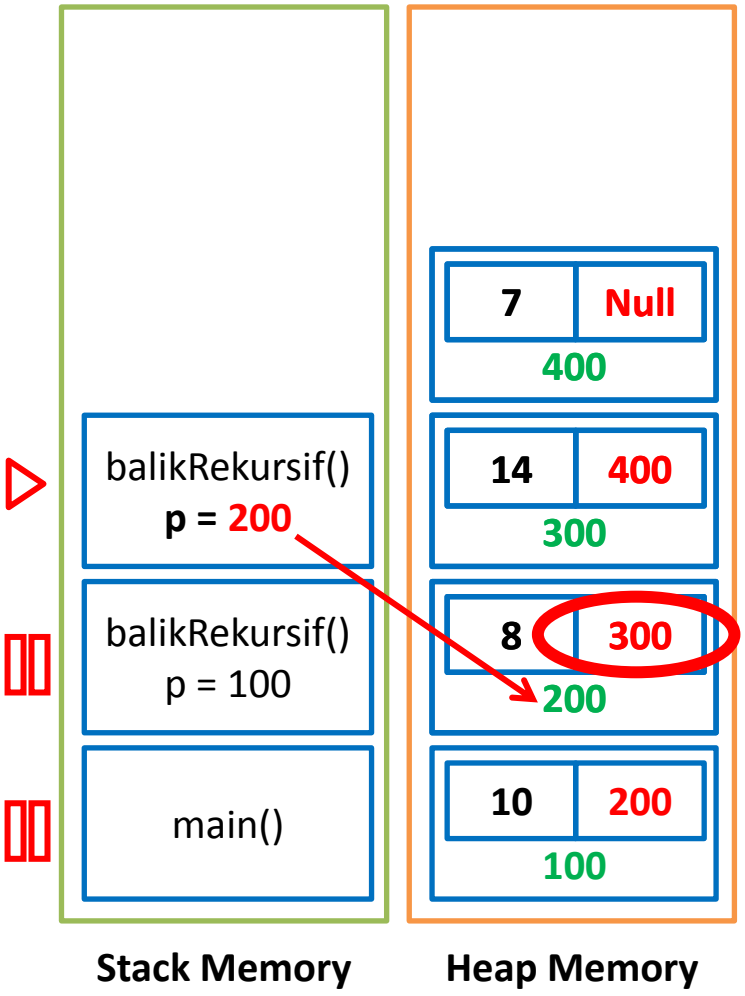


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

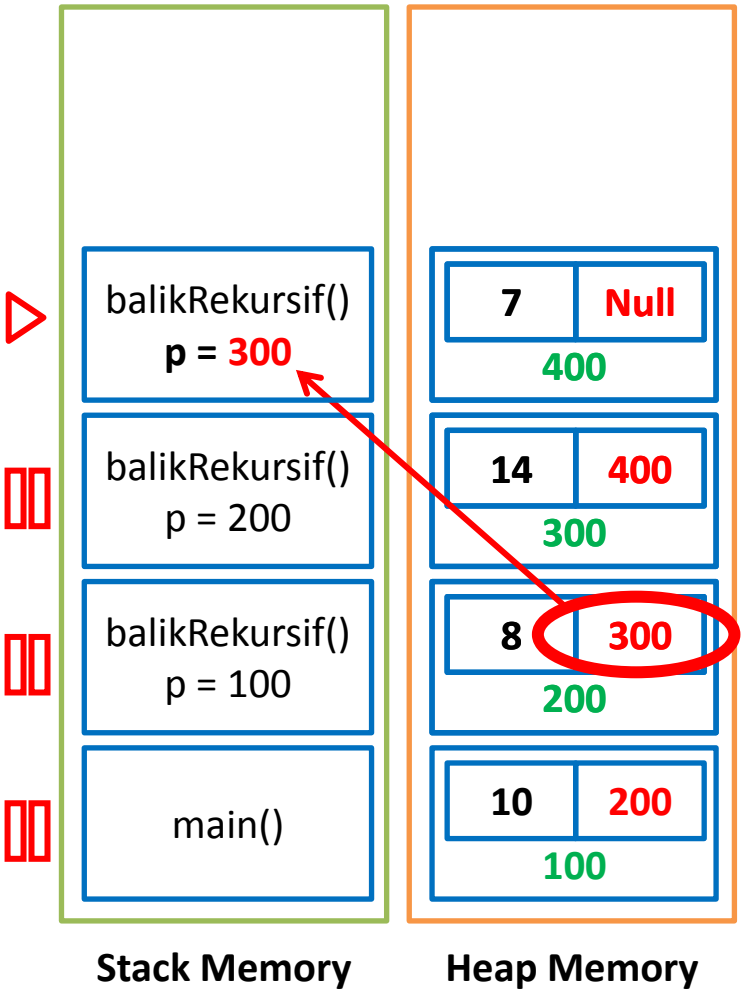


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

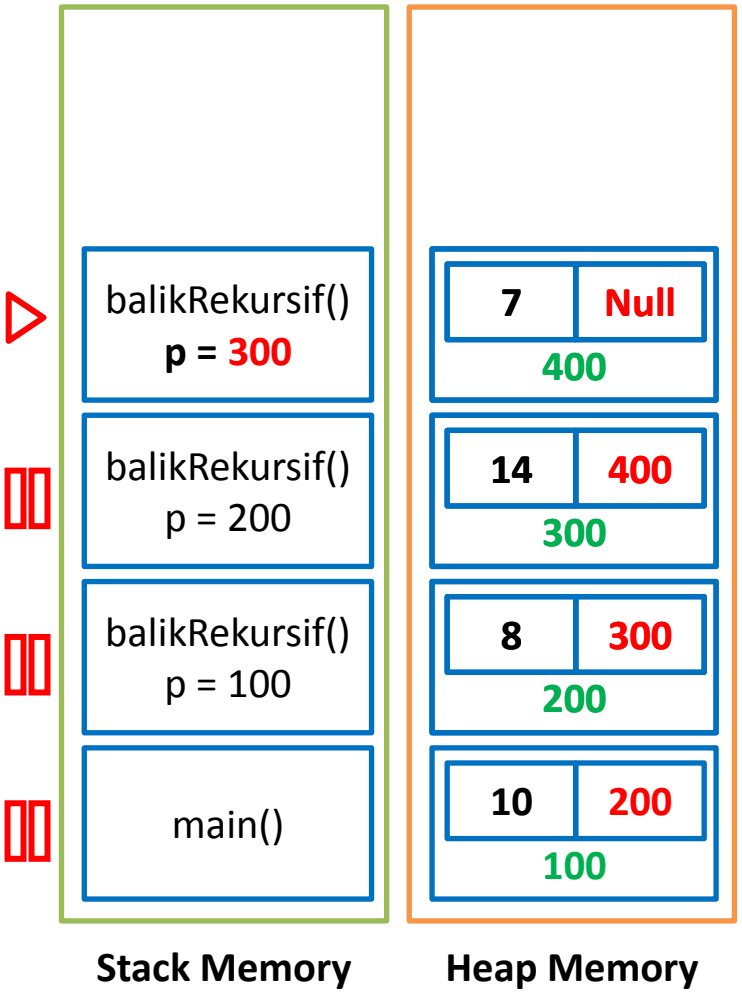


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

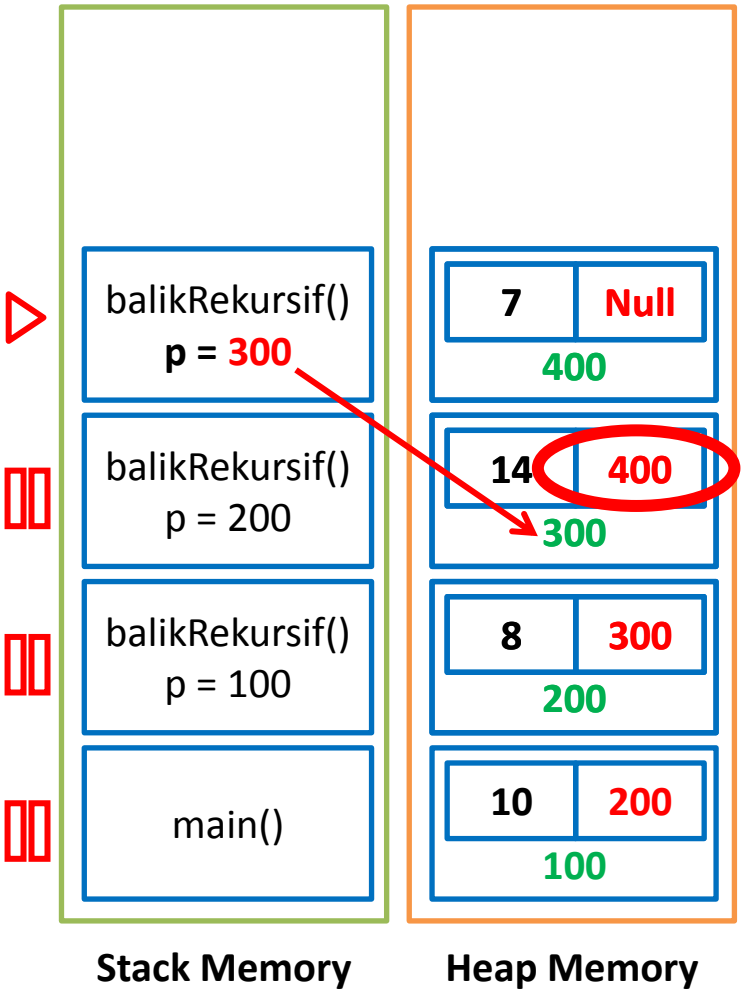


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

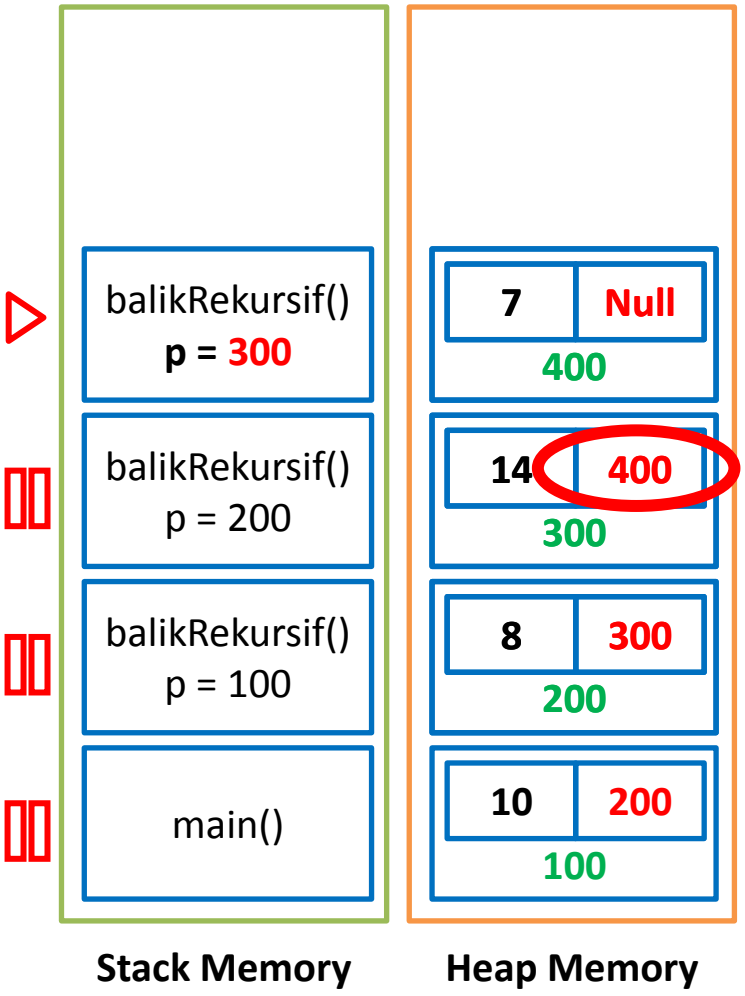


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

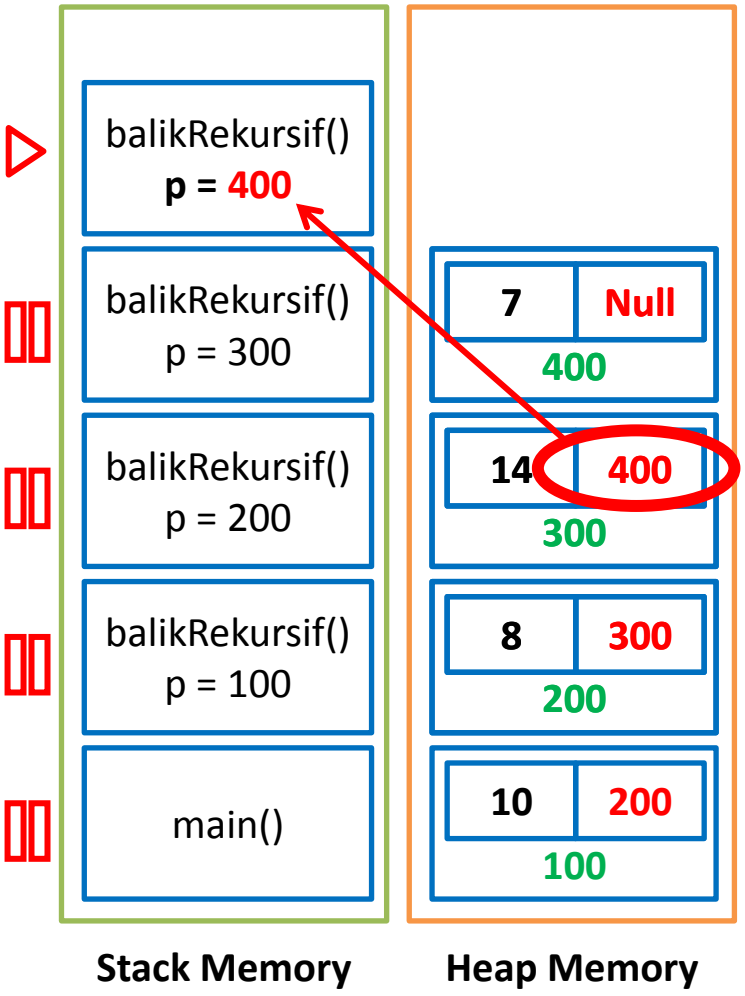


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

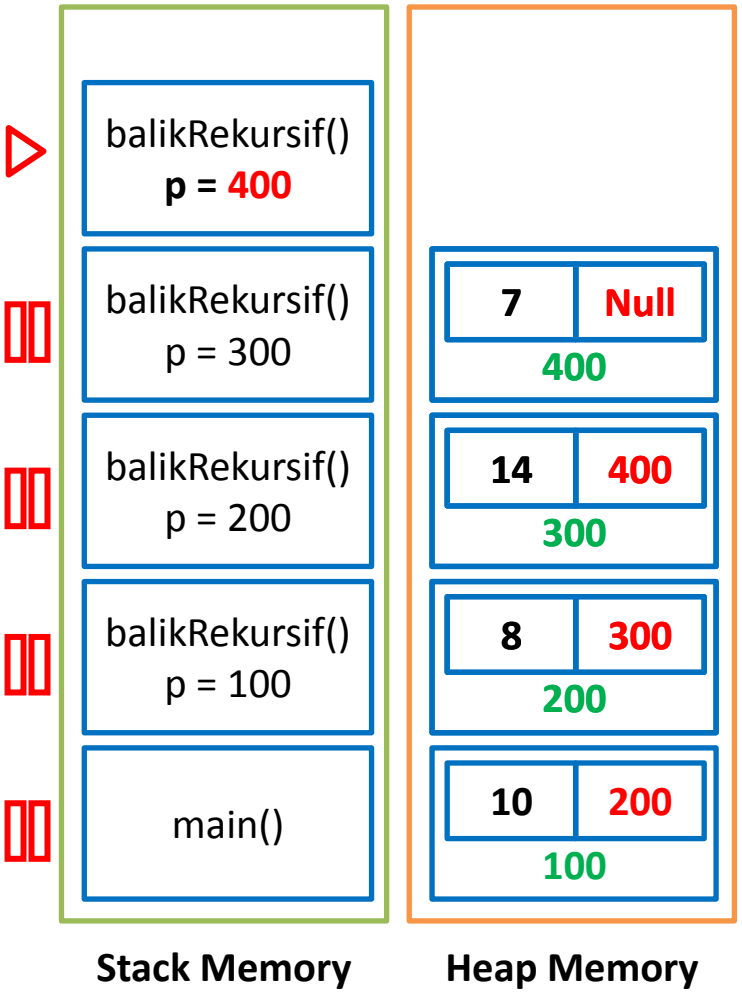


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

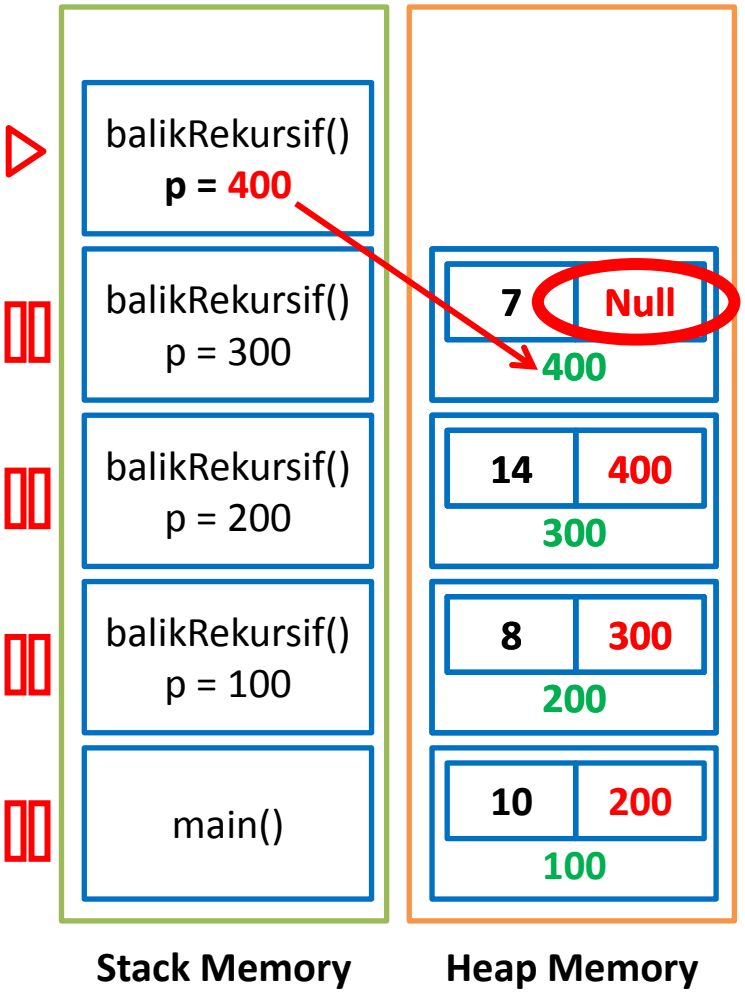


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=100
}
```

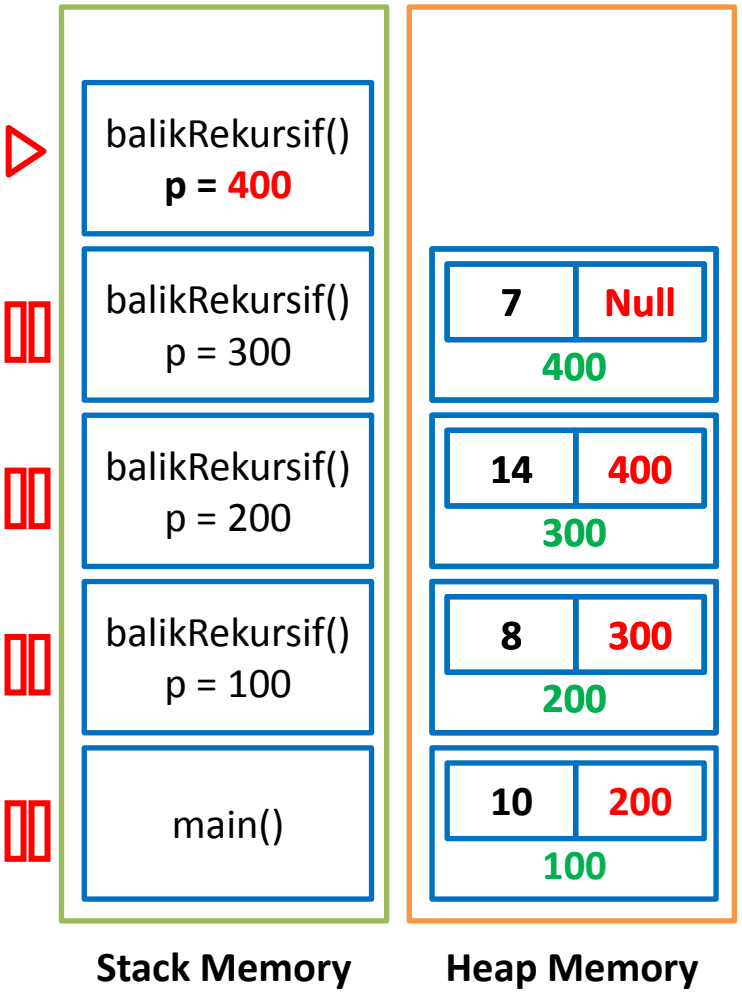


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

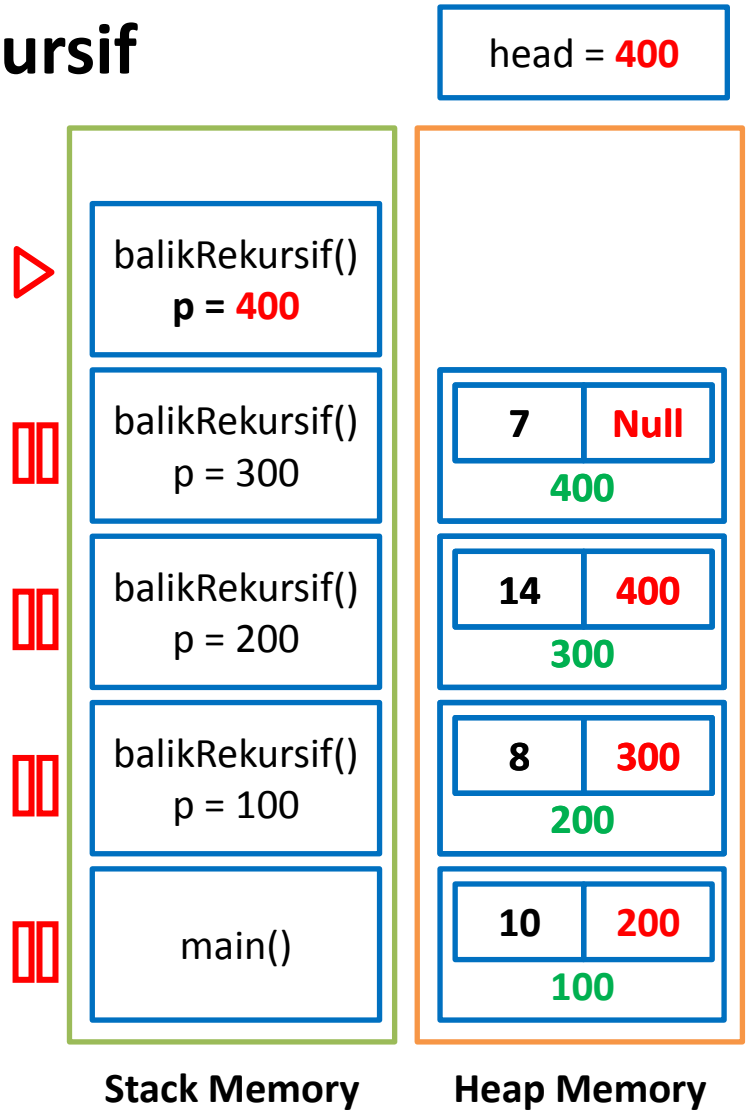


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

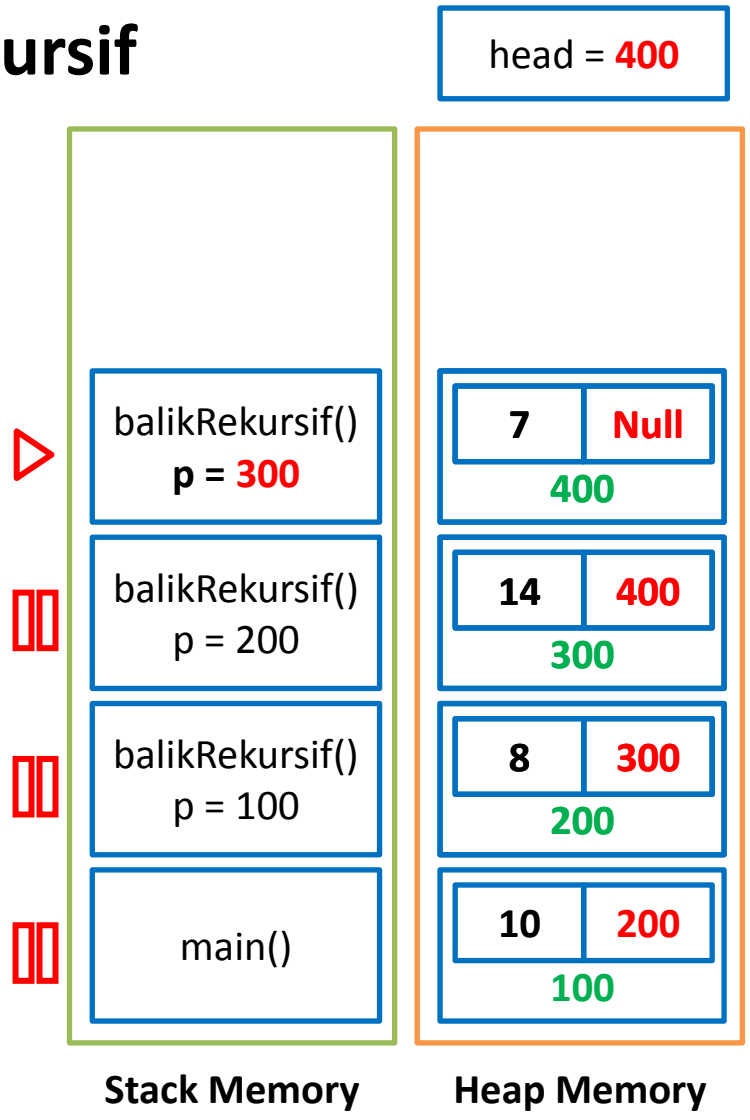


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

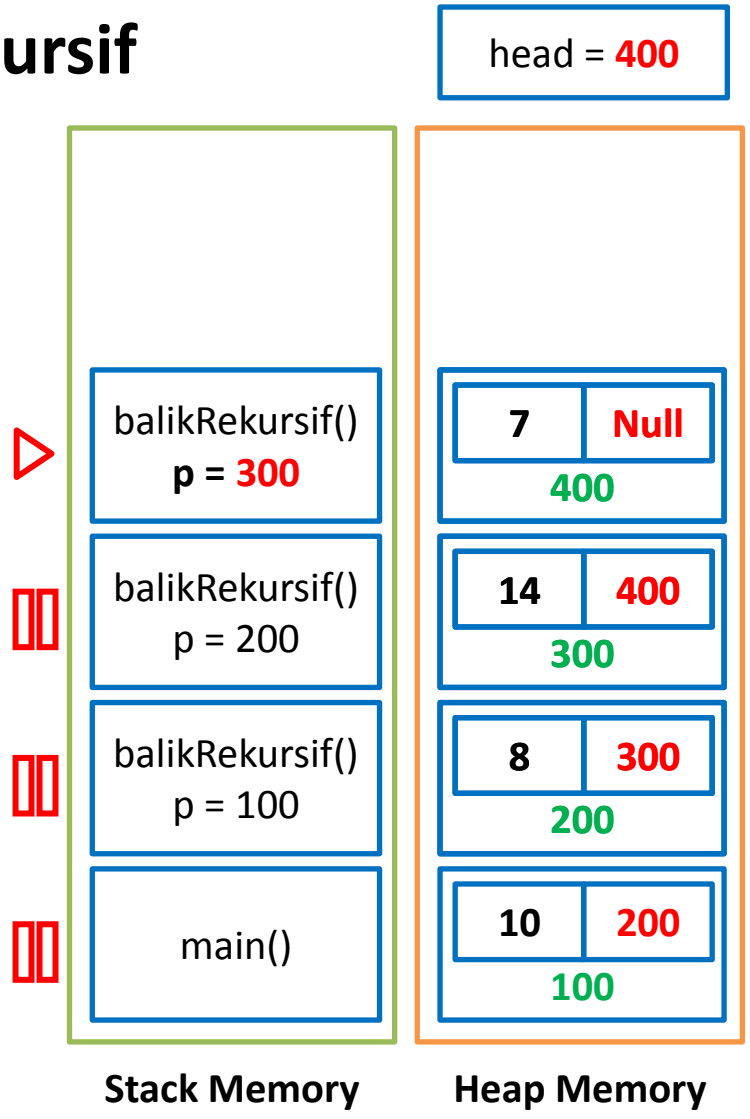


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

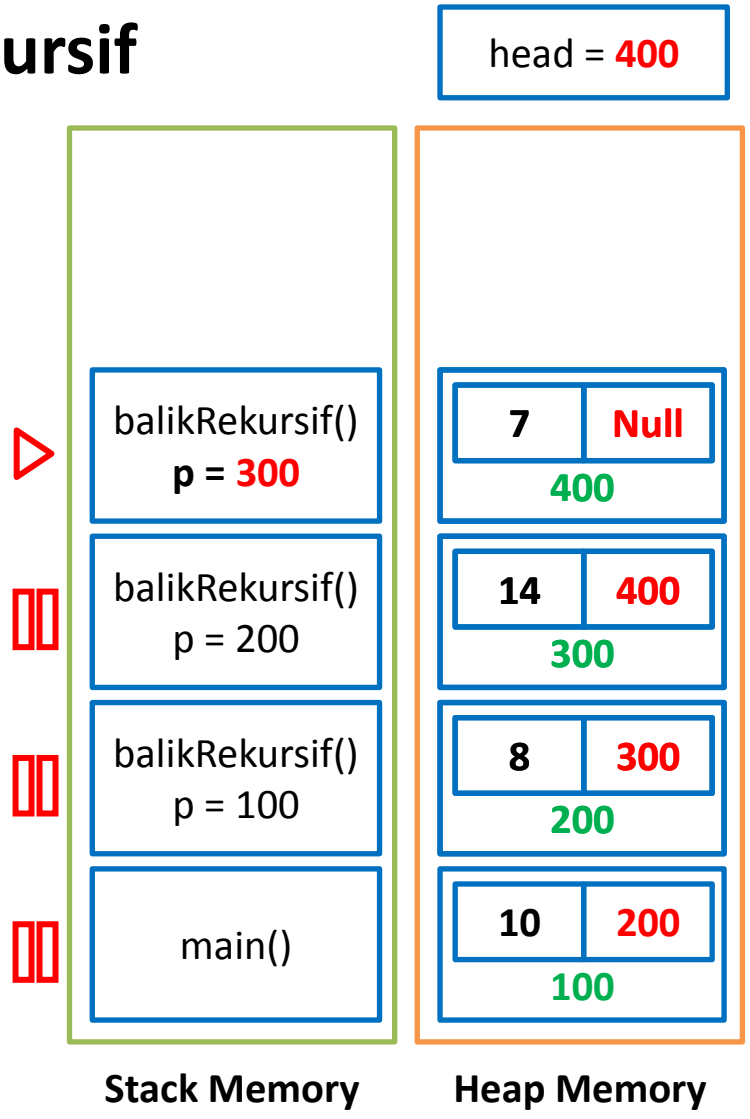


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

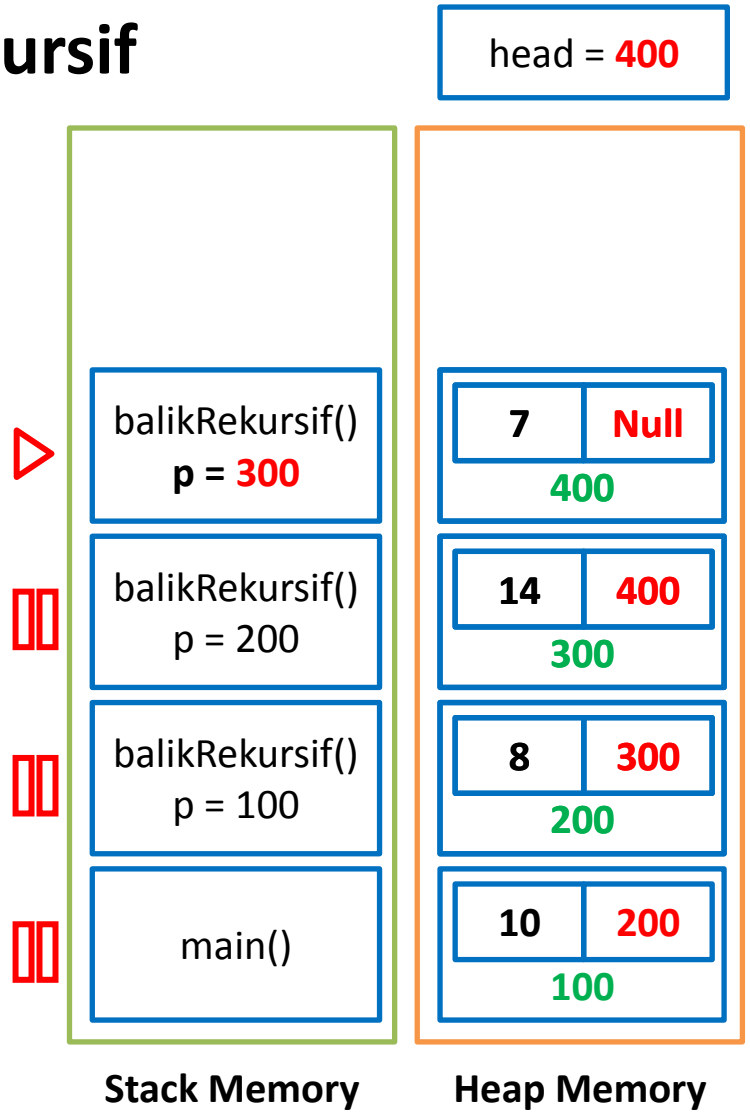


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

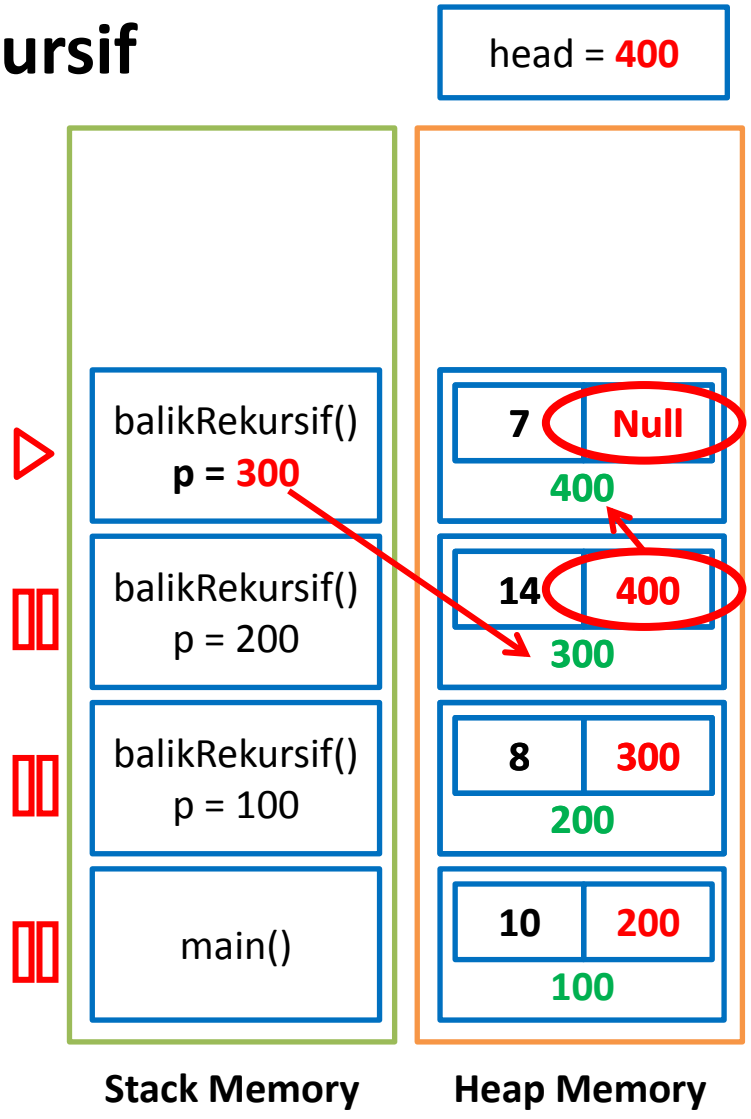


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

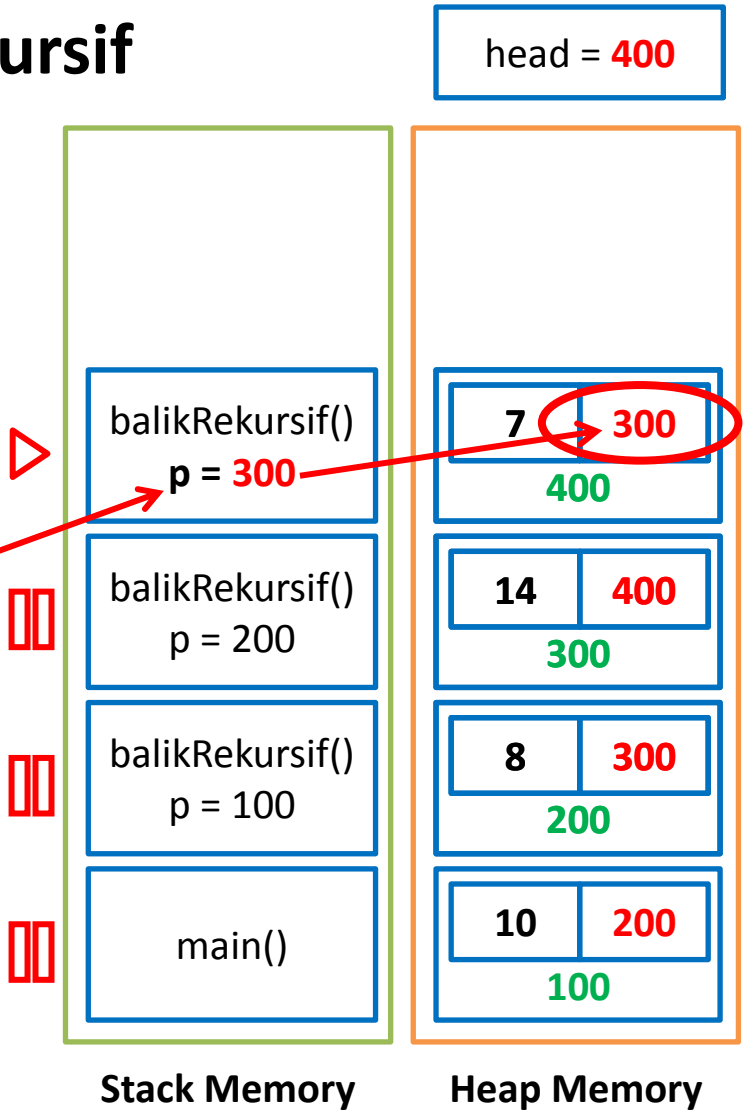
sudah dijalankan sebelumnya



Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        sudah dijalankan sebelumnya
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

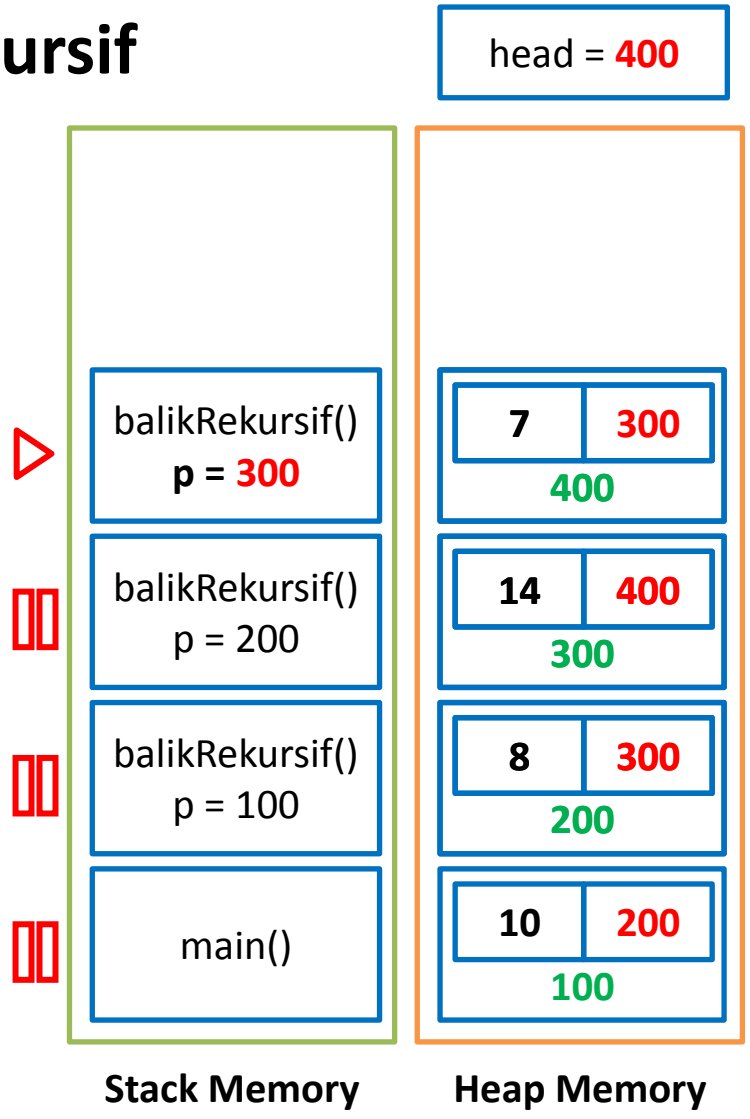


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

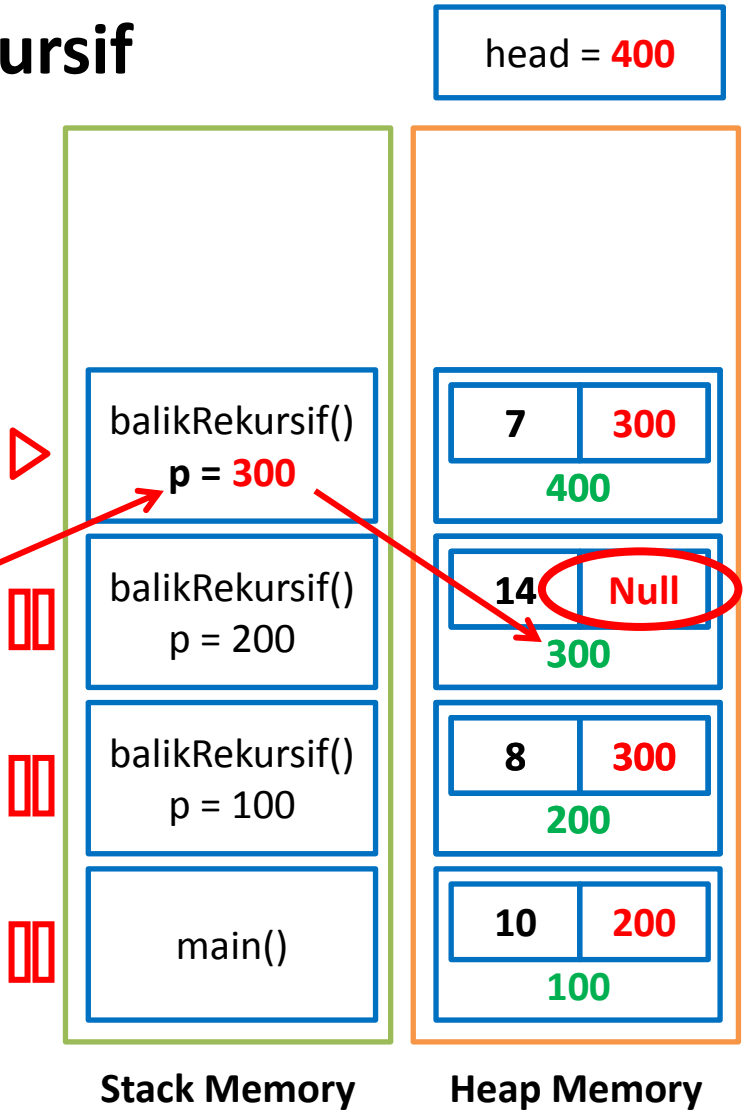


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

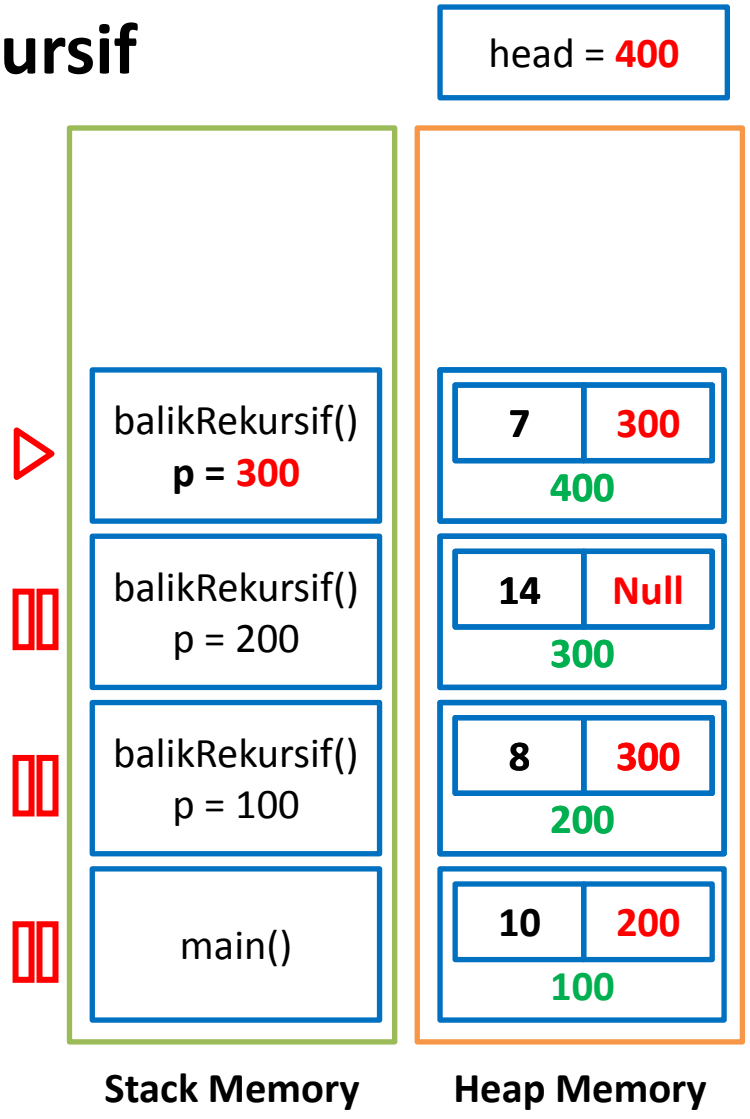


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

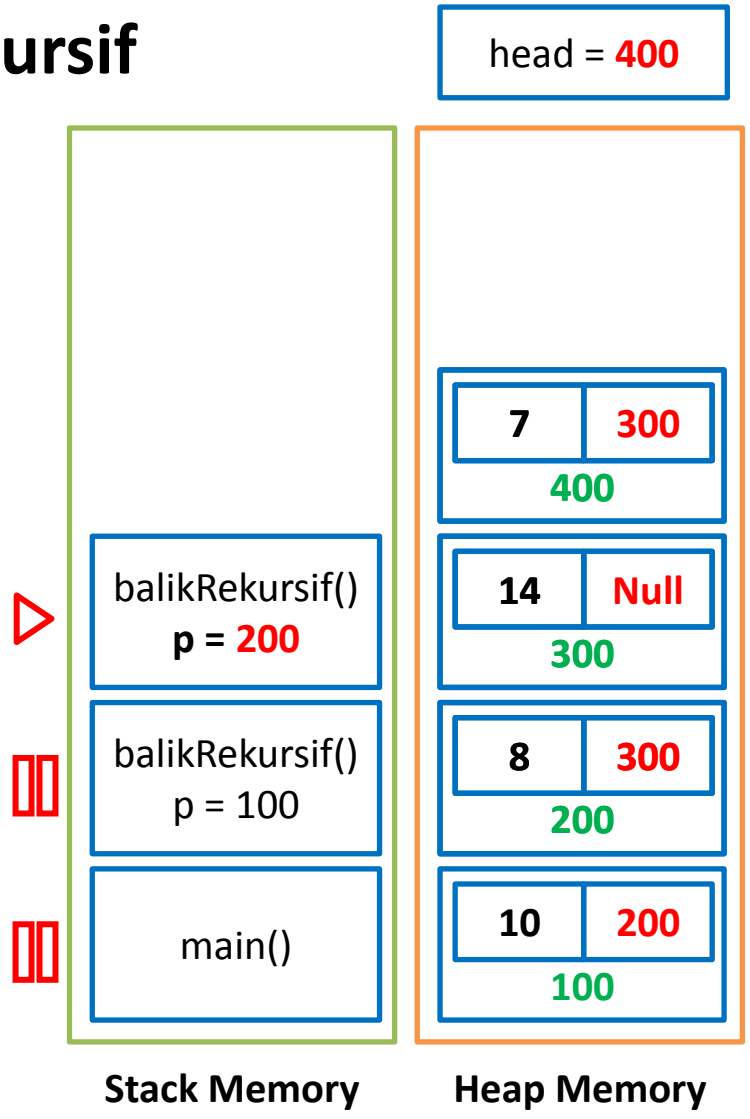


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

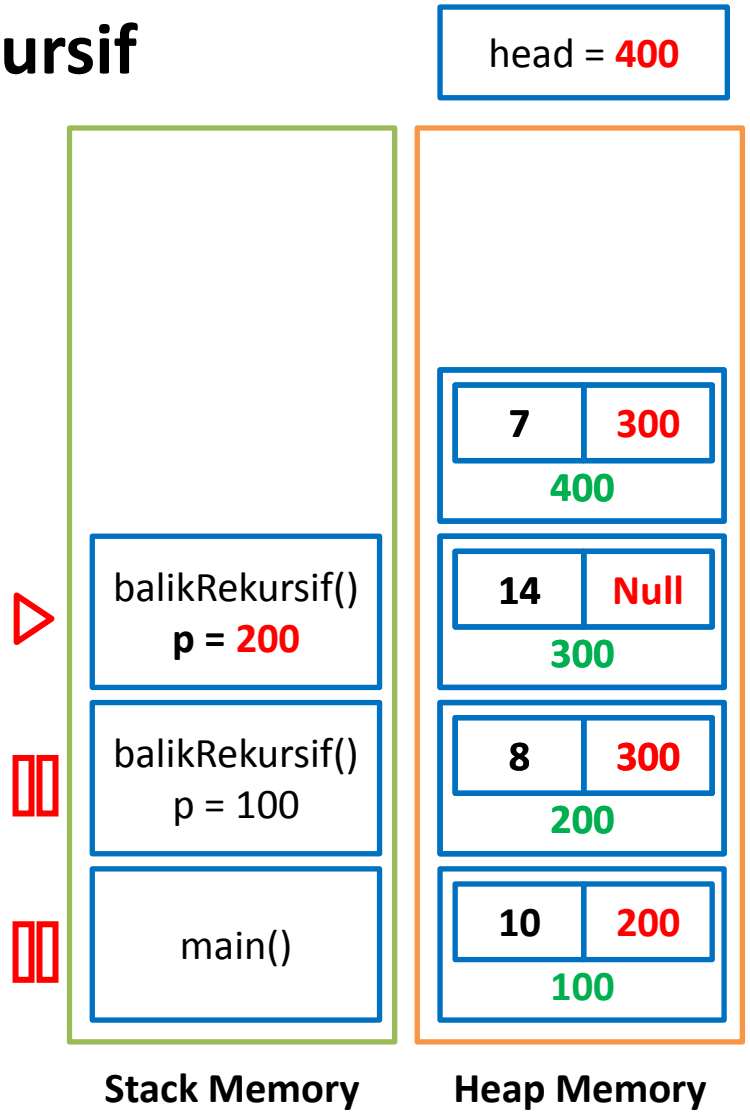


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

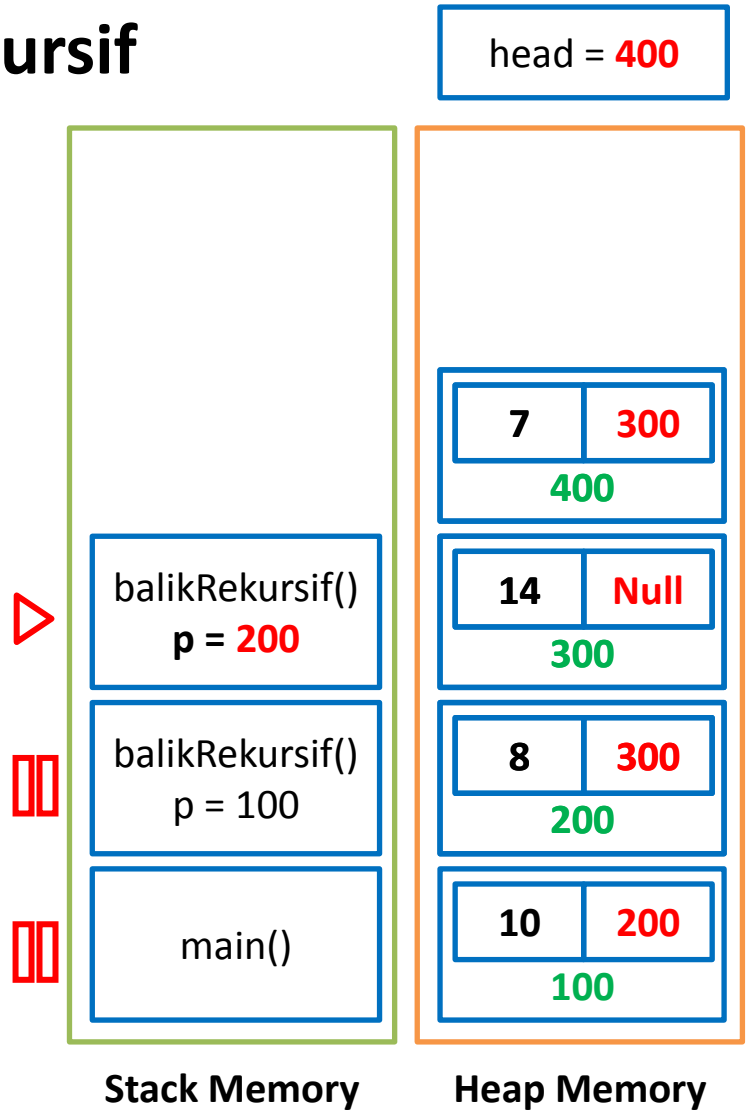


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

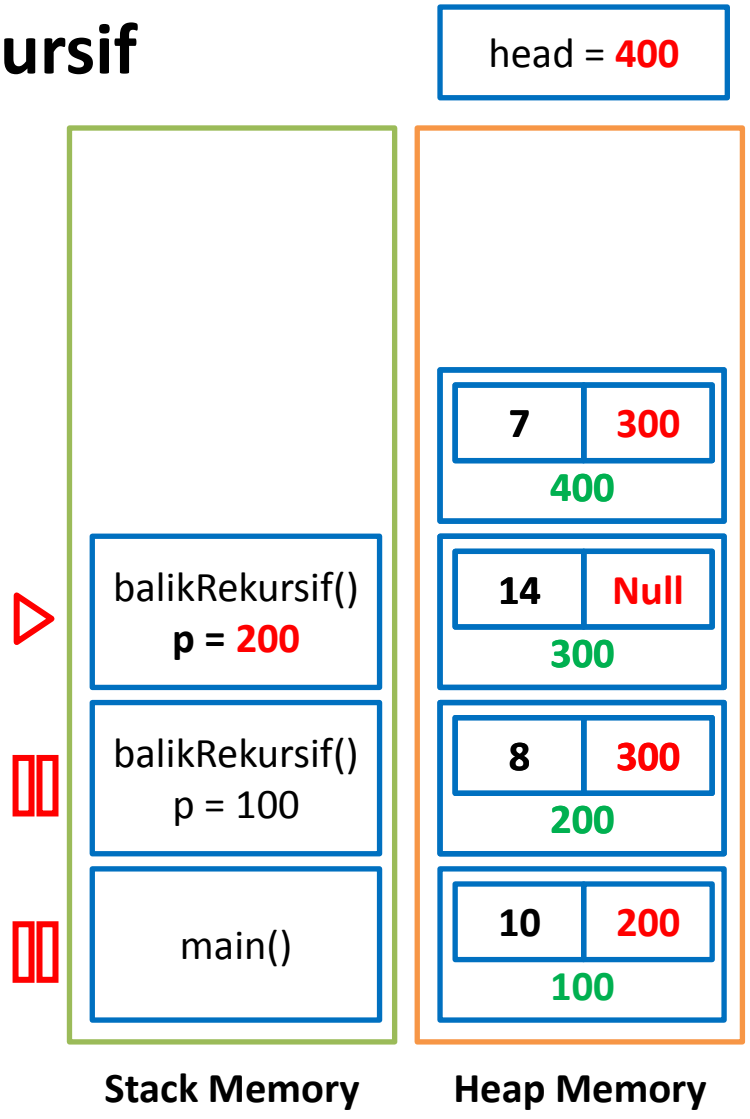


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya



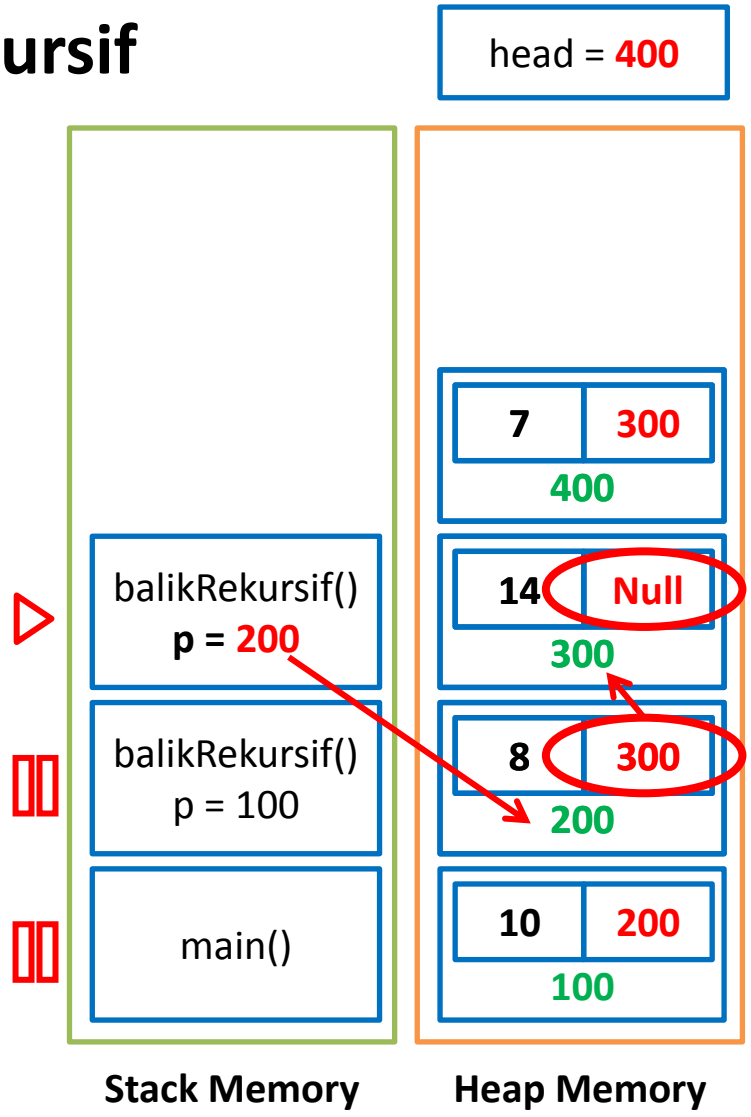
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

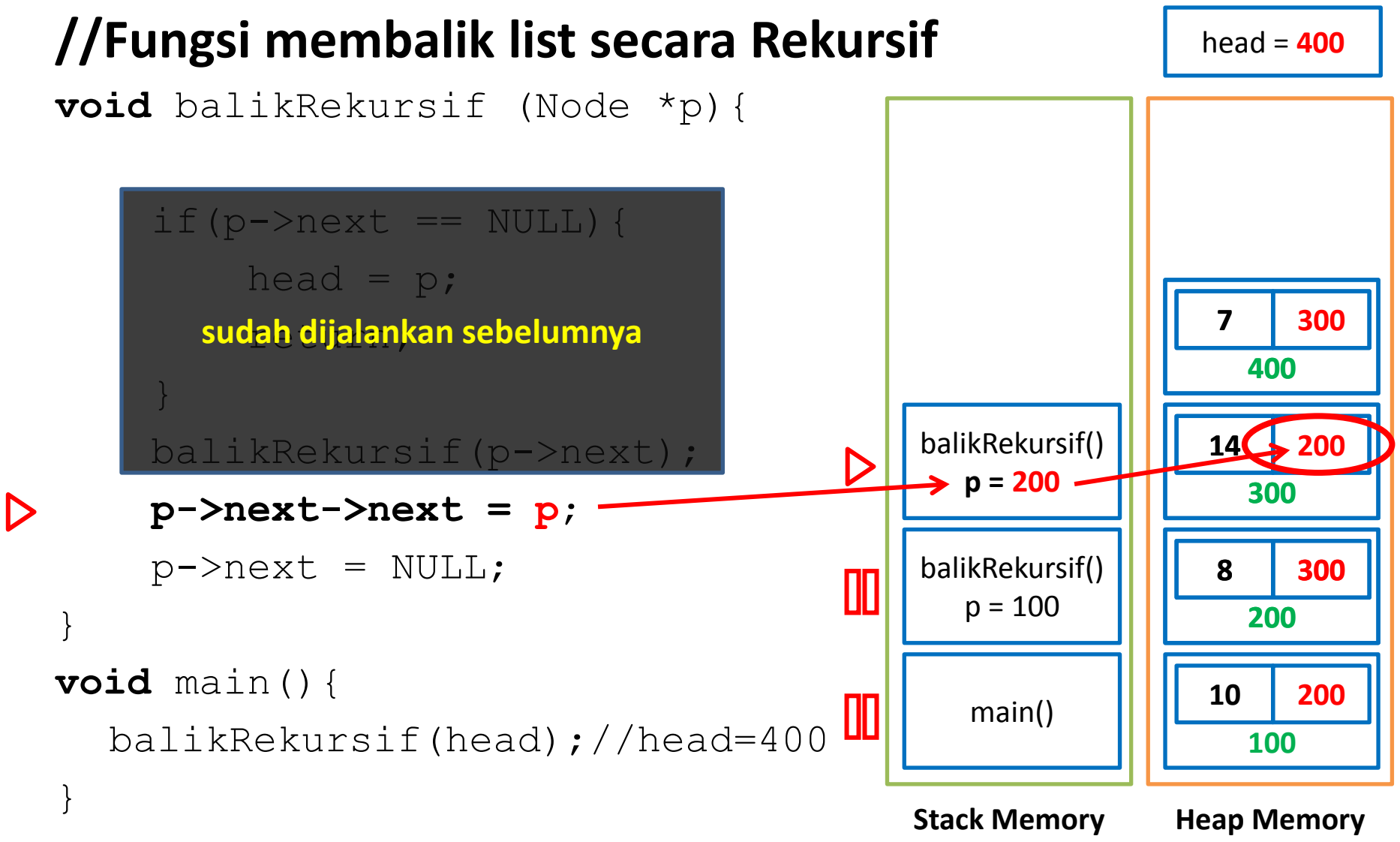


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

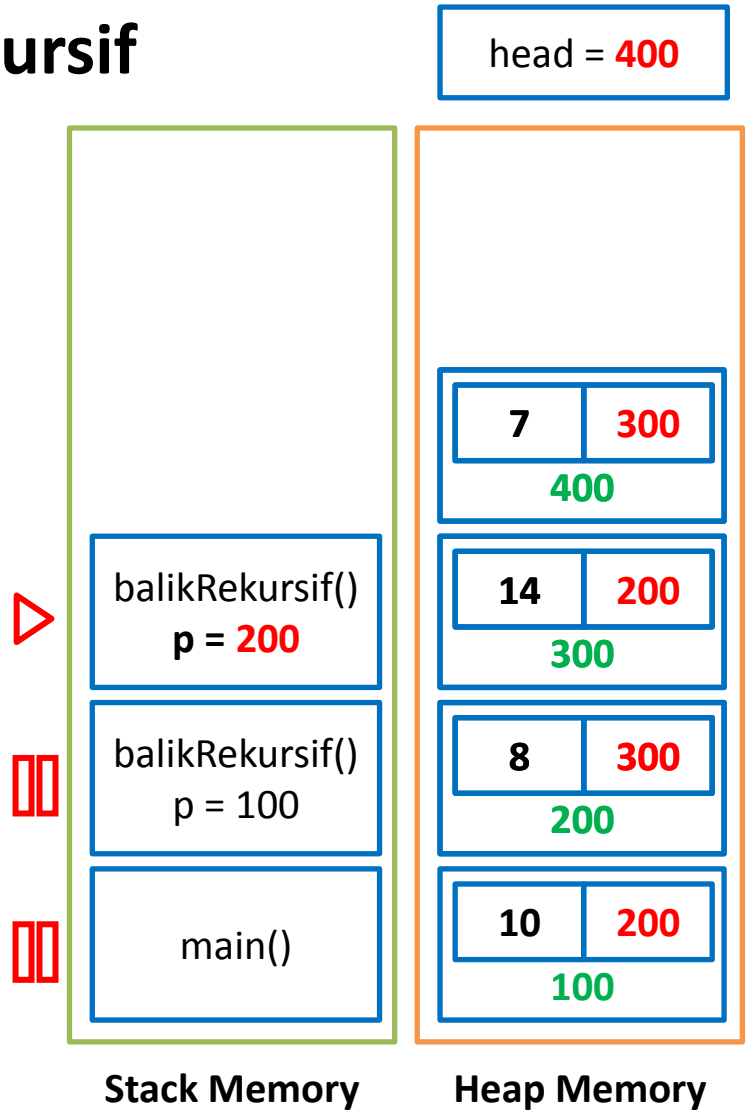


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

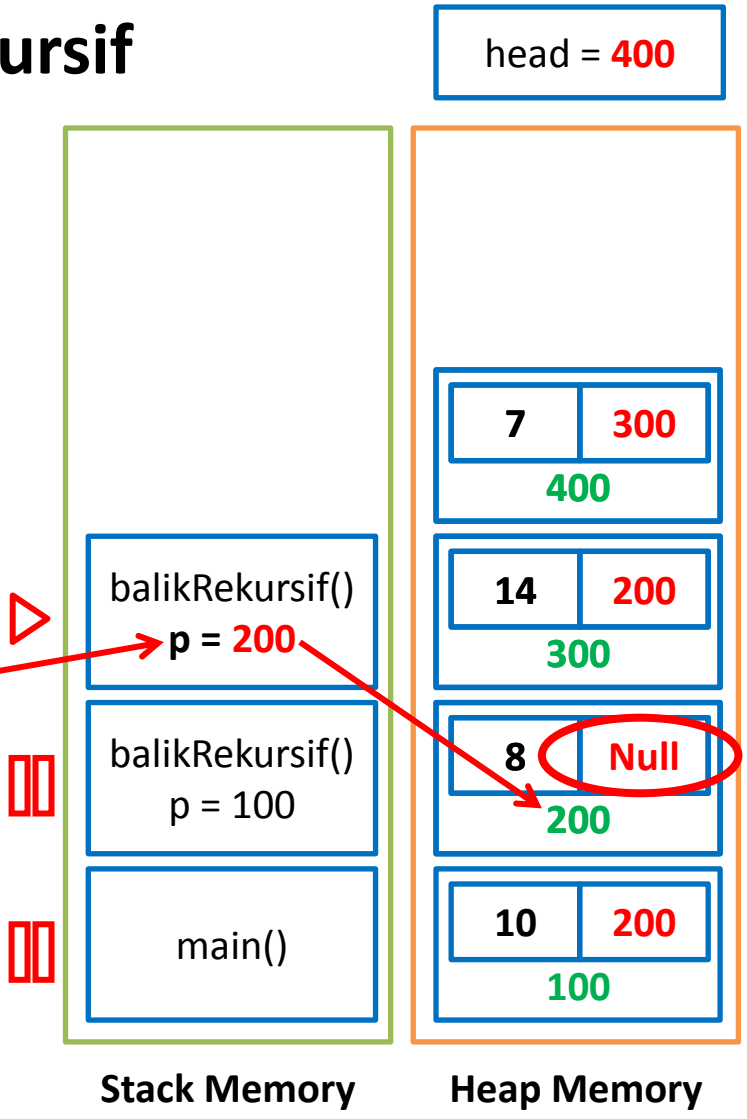


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

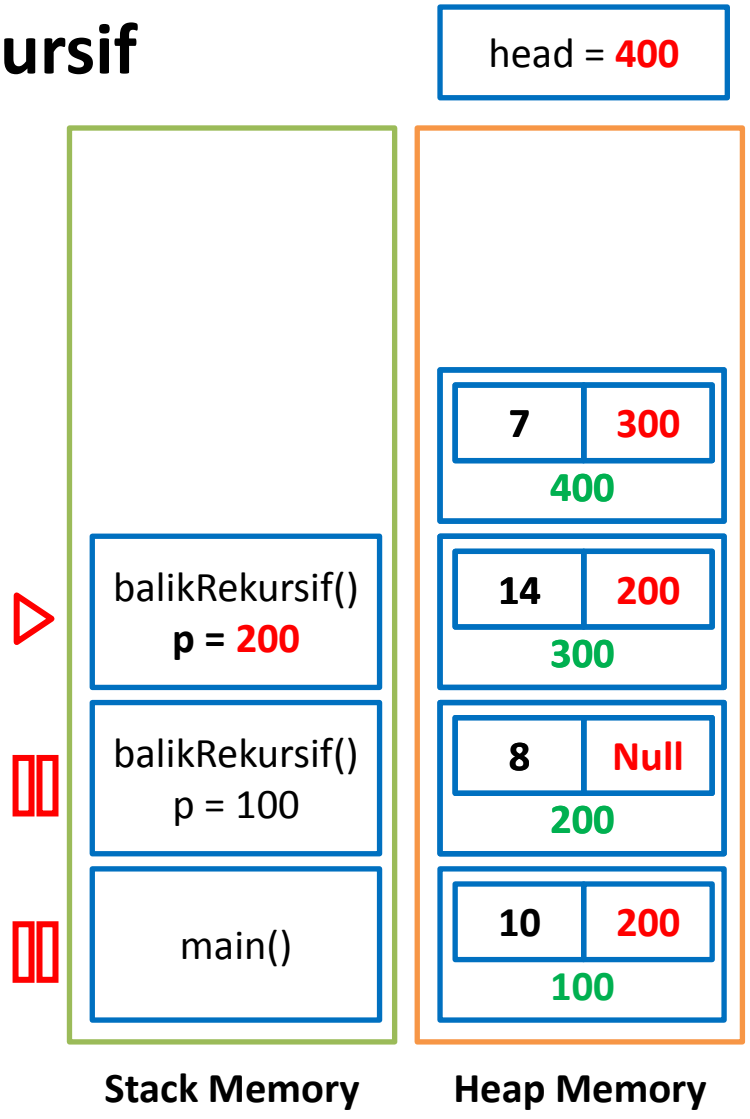


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

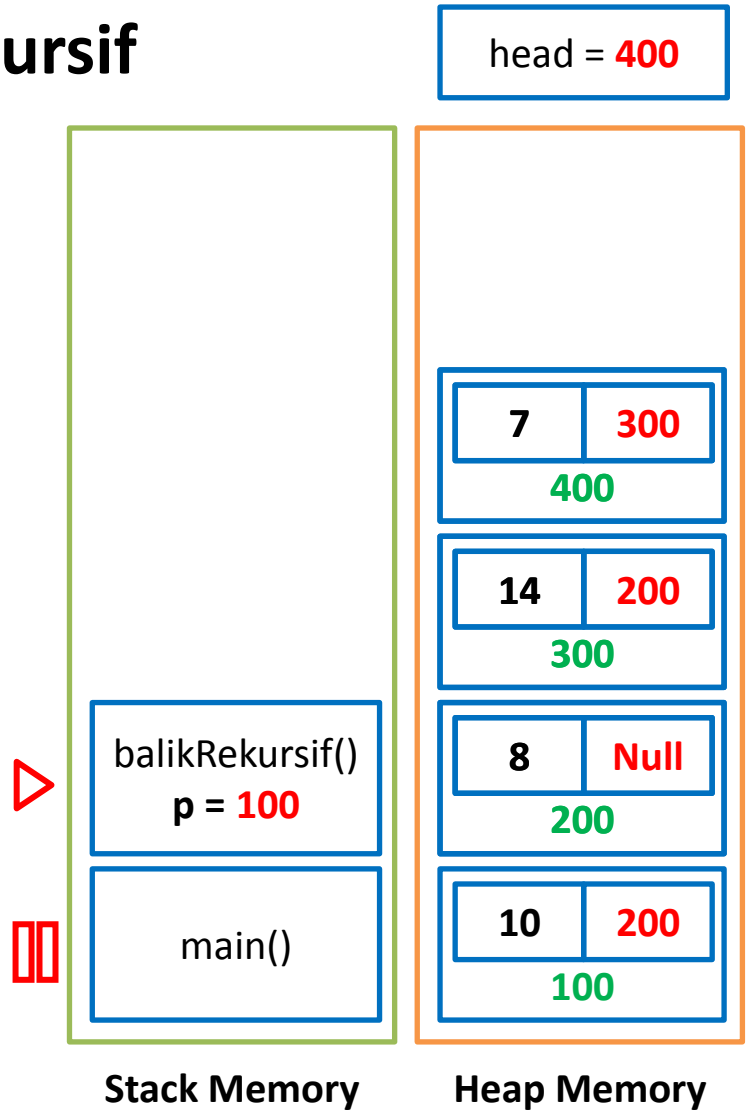


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

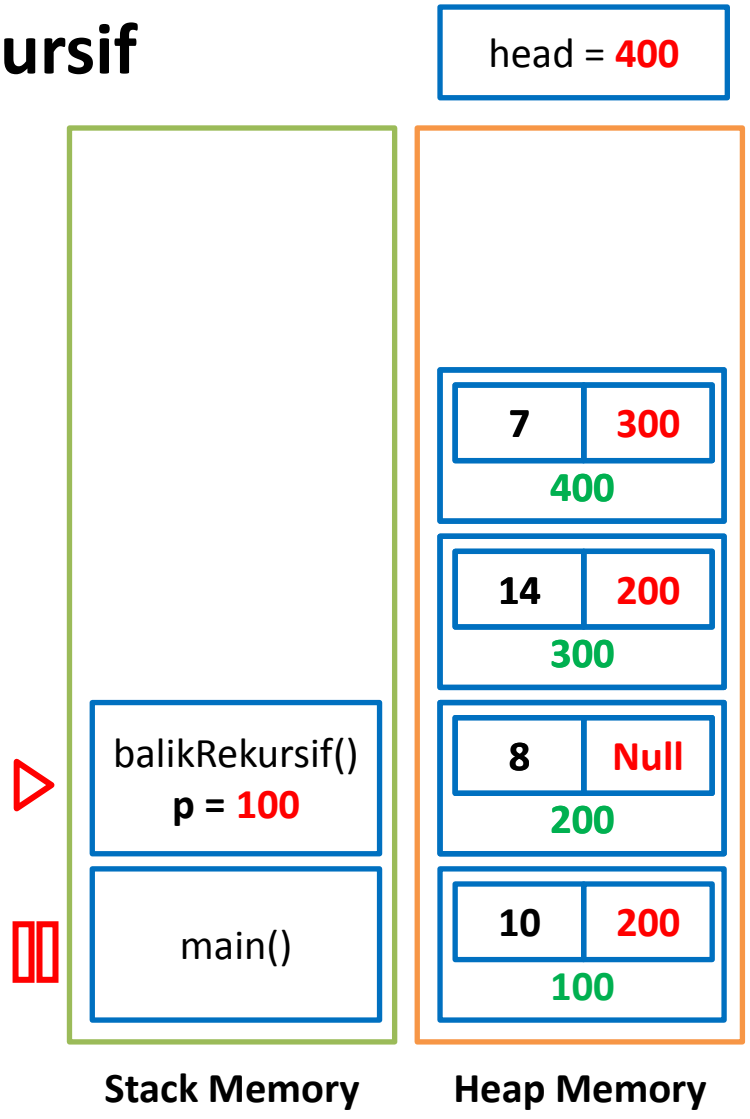


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```



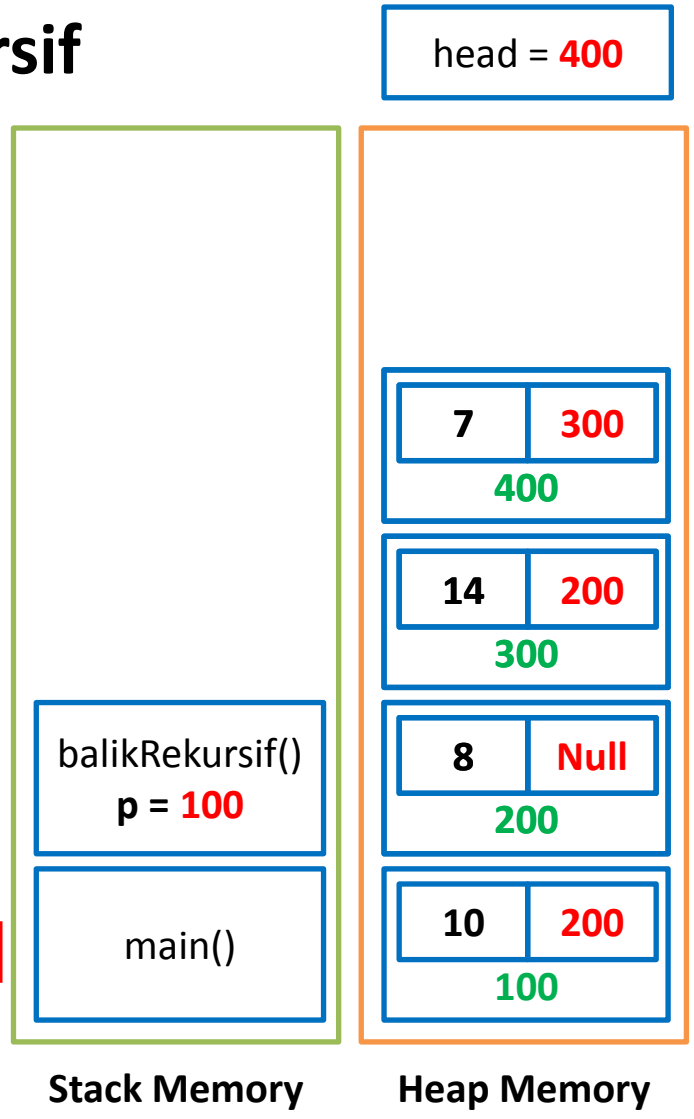
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya



Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

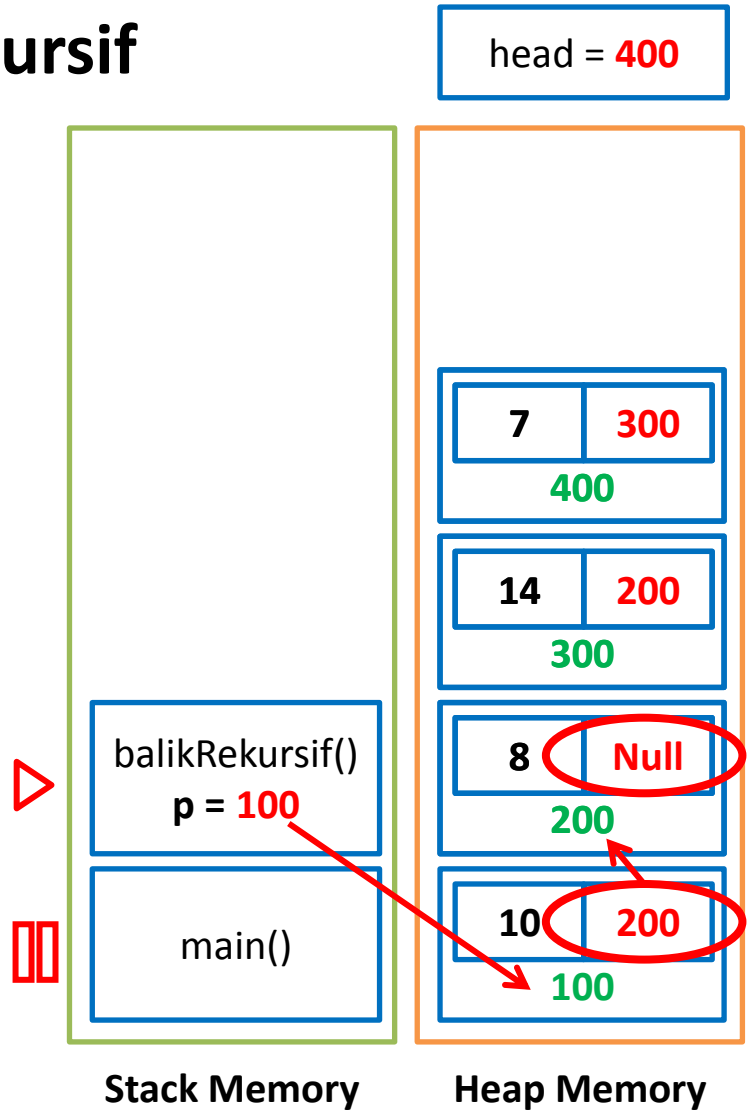
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya



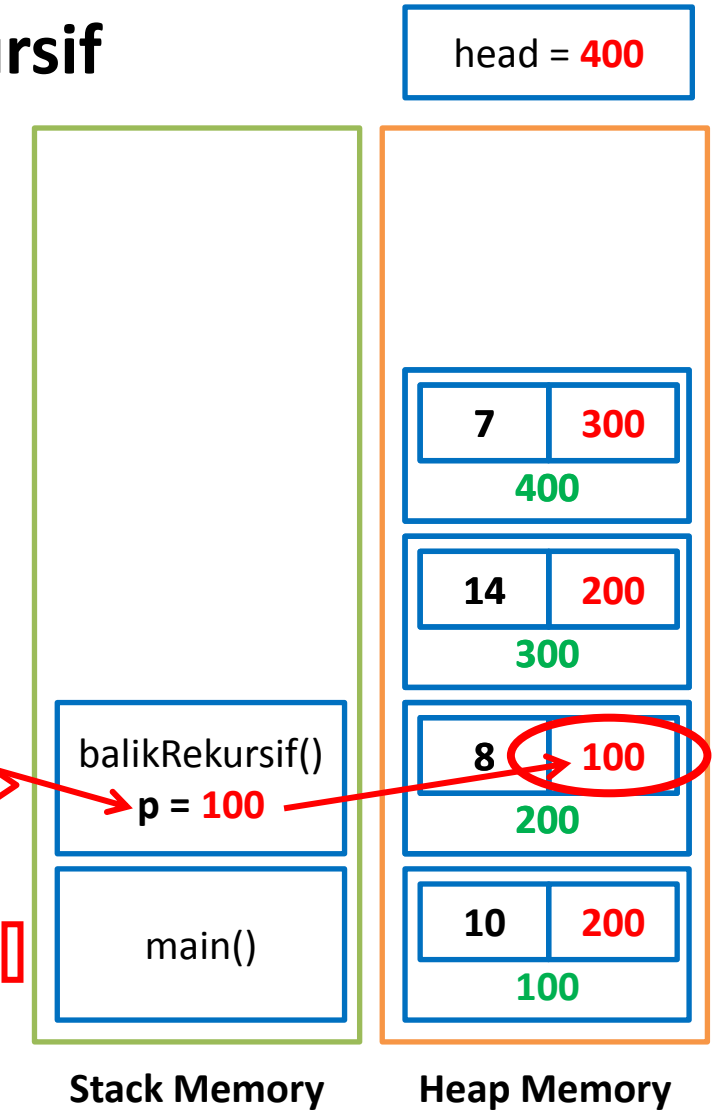
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

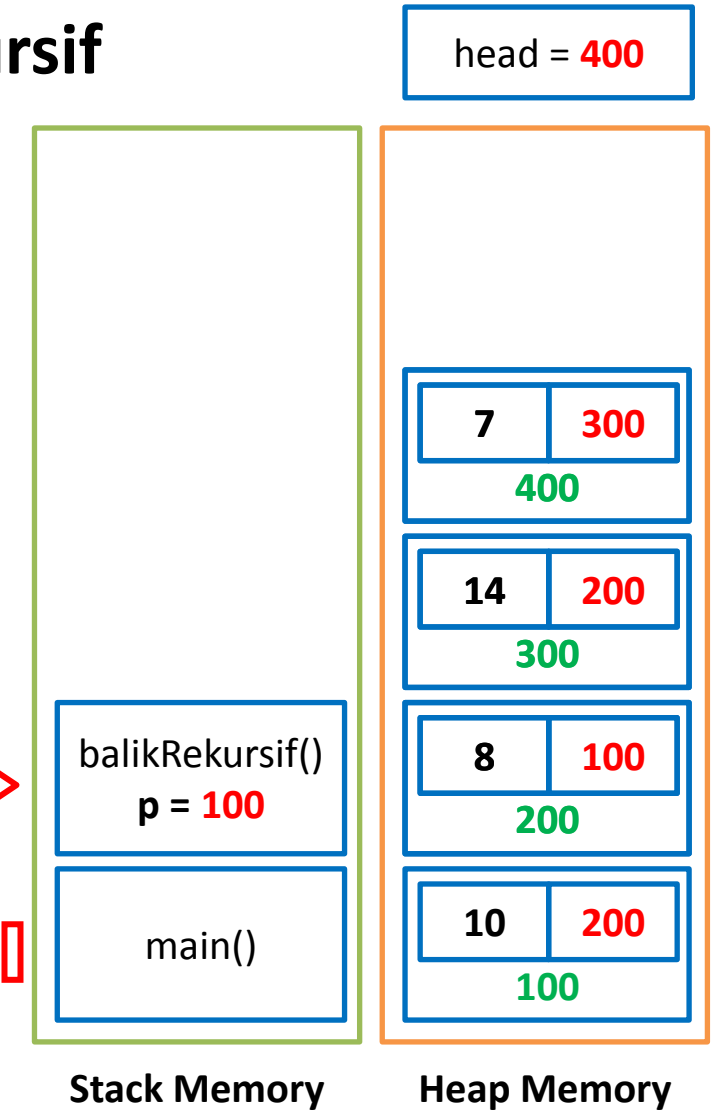


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya



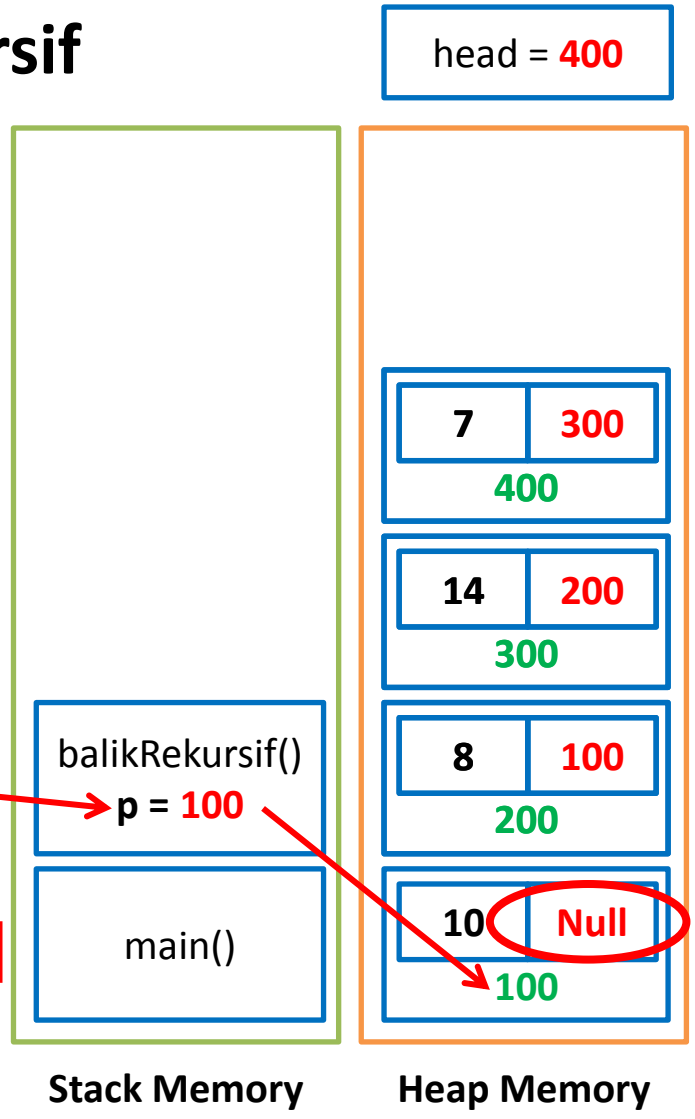
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

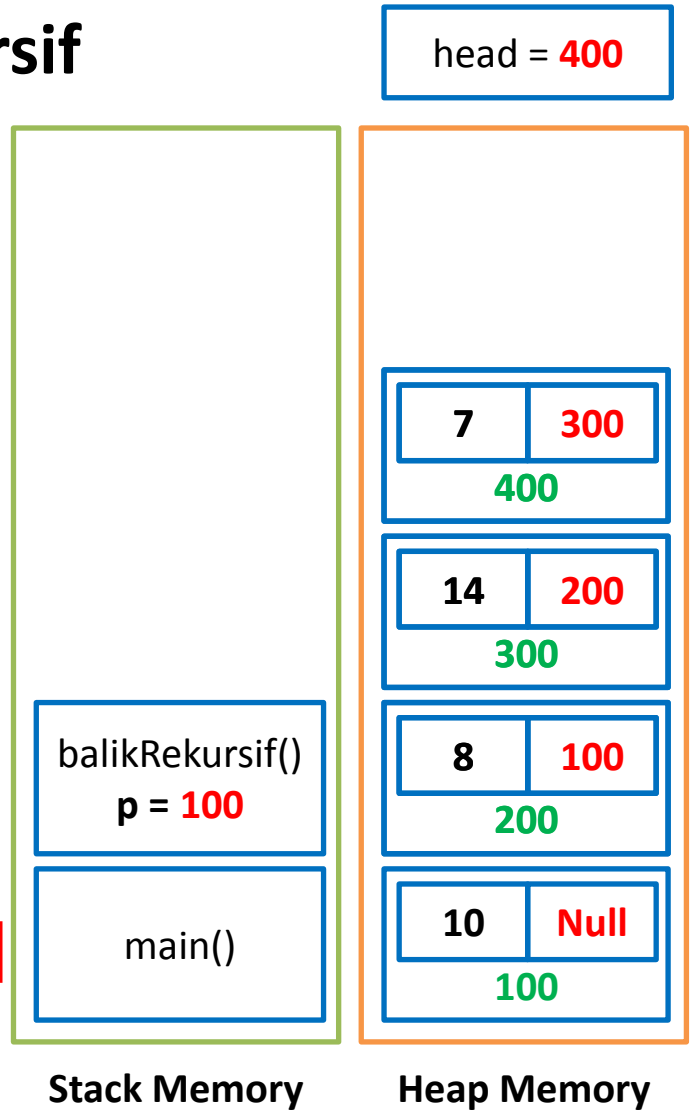


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){
    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
   balikRekursif(head);//head=400
}
```

sudah dijalankan sebelumnya

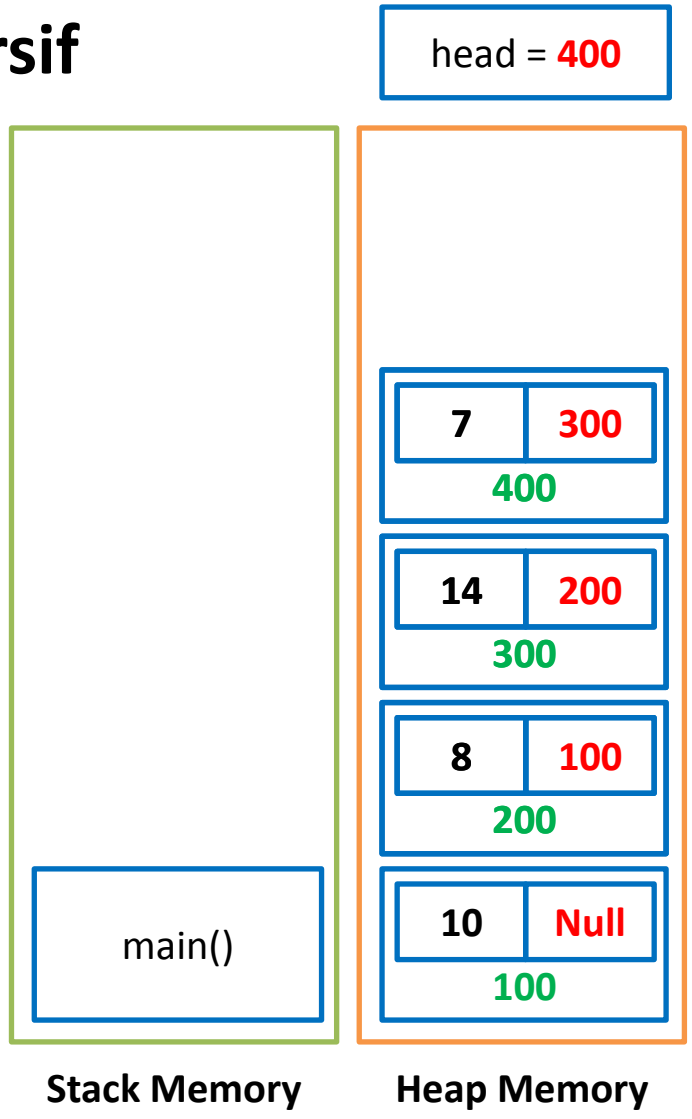


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```



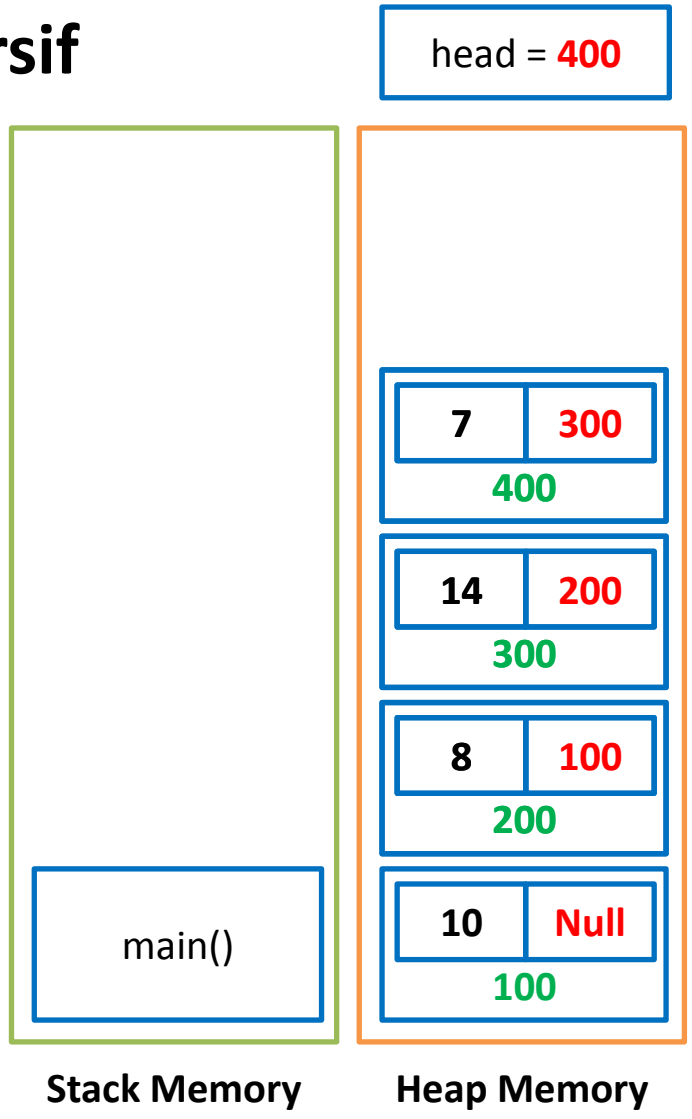
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

     if(p->next == NULL){
          head = p;
          return;
     }
     balikRekursif(p->next);
     p->next->next = p;
     p->next = NULL;
}
void main(){
     balikRekursif(head);//head=400
}
```

fungsi sudah dijalankan sebelumnya



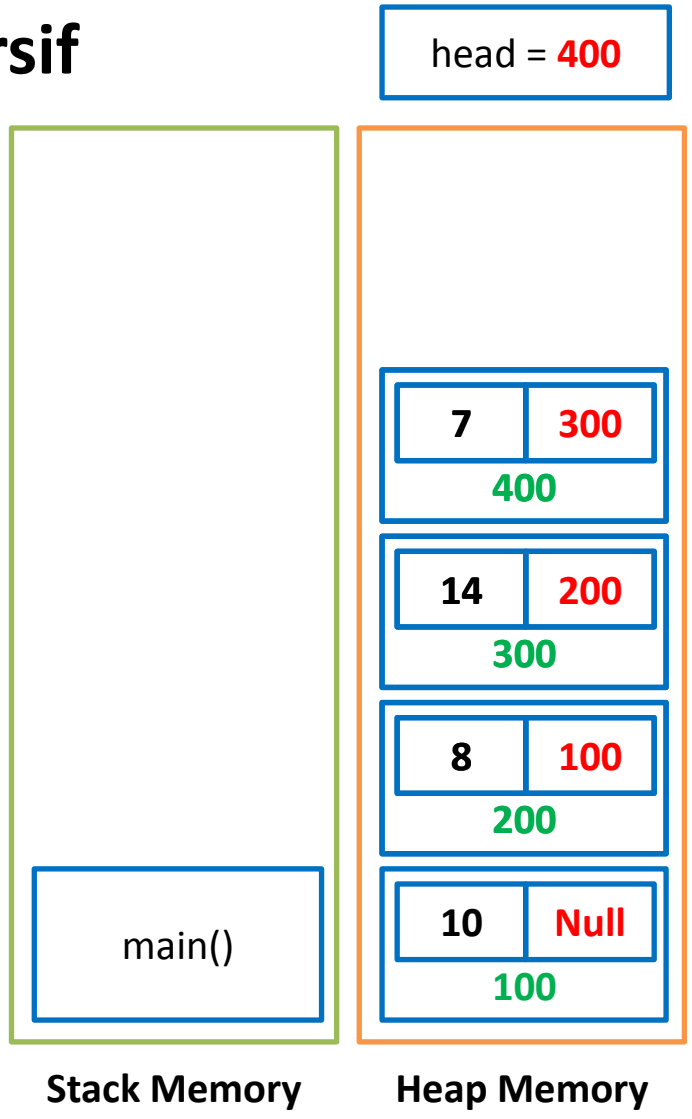
Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
    balikRekursif(head);//head=400
}
```

fungsi sudah dijalankan sebelumnya

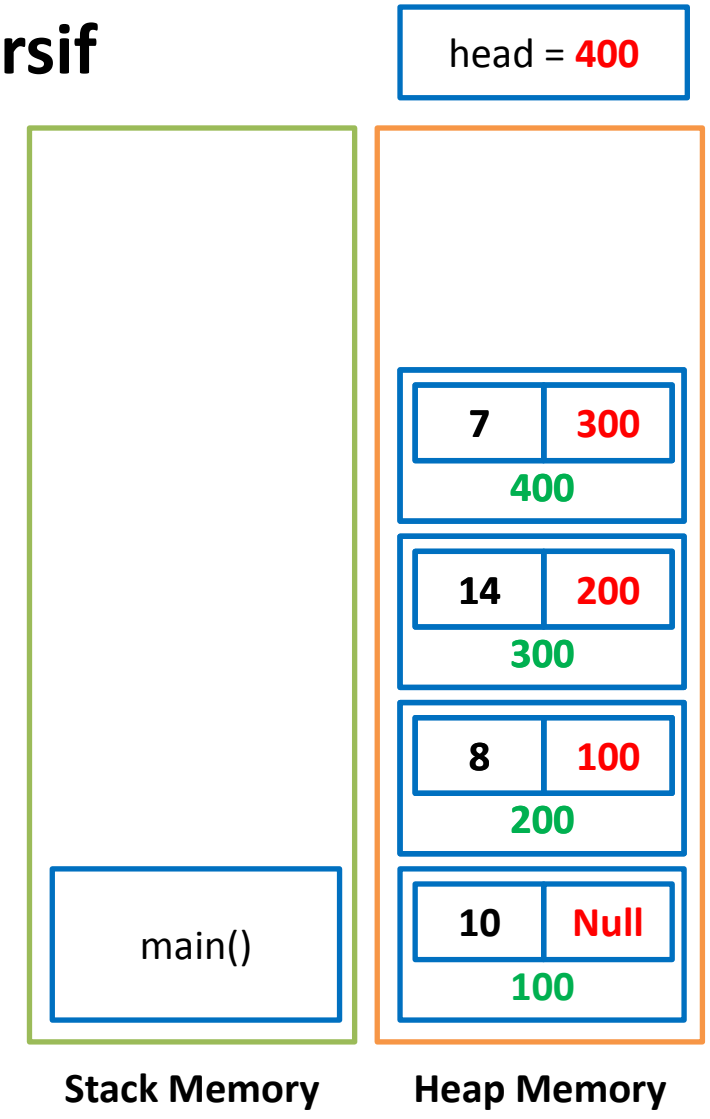


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```

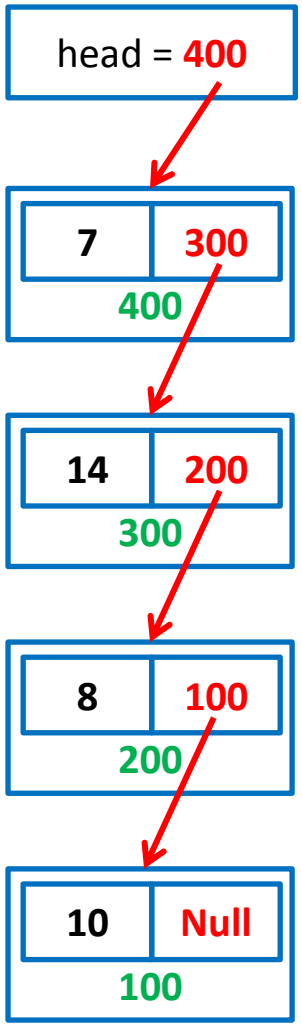


Studi Kasus: Membalik List secara Rekursif

//Fungsi membalik list secara Rekursif

```
void balikRekursif (Node *p){

    if(p->next == NULL){
        head = p;
        return;
    }
    balikRekursif(p->next);
    p->next->next = p;
    p->next = NULL;
}
void main(){
  balikRekursif(head);//head=400
}
```



Sekian

TERIMAKASIH