LIST REKURSIF

Danang Wahyu Utomo
danang.wu@dsn.dinus.ac.id
+6285 740 955 623

RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	ADT Stack
2	ADT Queue
3	List Linear
4	List Linear
5	List Linear
6	Representasi Fisik List Linear
7	Variasi List Linear
8	**Ujian Tengah Semester**

W	Pokok Bahasan
9	Variasi List Linear
10	Double Linked List
11	Stack dengan Representasi List
12	Queue dengan Representasi List
13	List Rekursif
14	Pohon dan Pohon Biner
15	Multi List
16	**Ujian Akhir Semester**

Konten

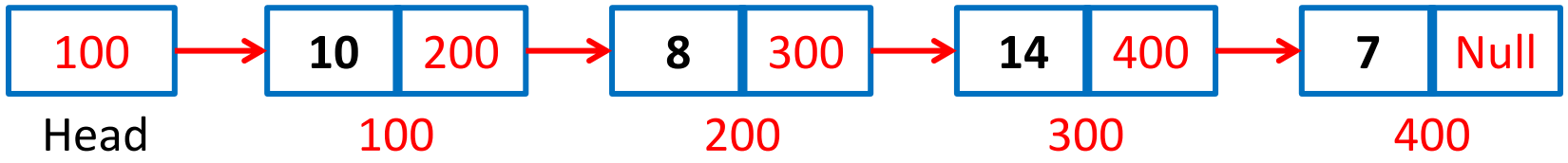
- Pendekatan Iteratif dan Rekursif
- Membalik List secara Iteratif
- Fungsi Rekursif pada Memory
- Membalik List secara Rekursif

Pendekatan Iteratif dan Rekursif

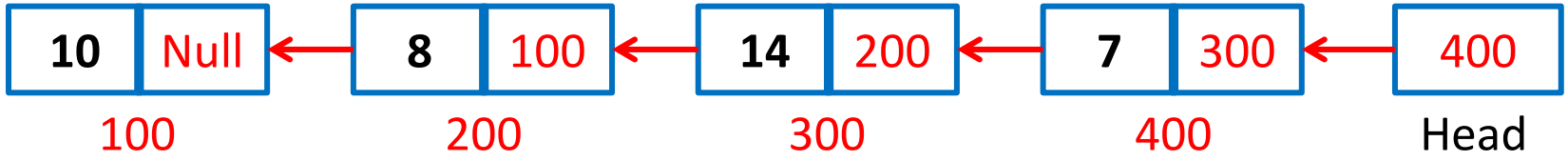
- Pendekatan iteratif menggunakan proses perulangan (loop) untuk menyelesaikan masalah
- Dalam konteks prosedural kita memiliki loop sebagai mekanisme untuk mengulang
- Suatu entitas disebut rekursif jika pada definisinya terkandung dirinya sendiri
- Program prosedural juga dapat bersifat rekursif
- Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri

Membalik List secara Iteratif

- Input



- Output



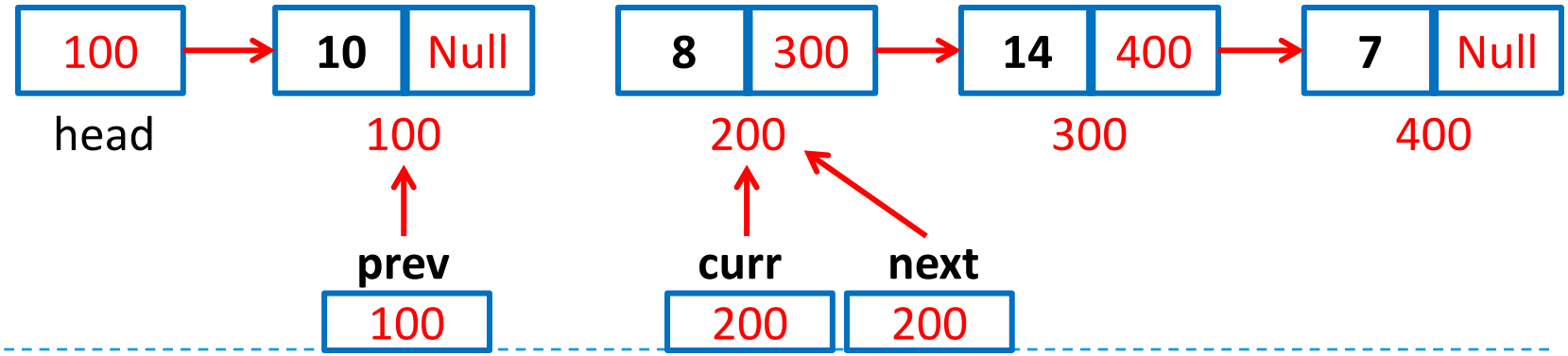
Membalik List secara Iteratif

- Diperlukan tiga 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



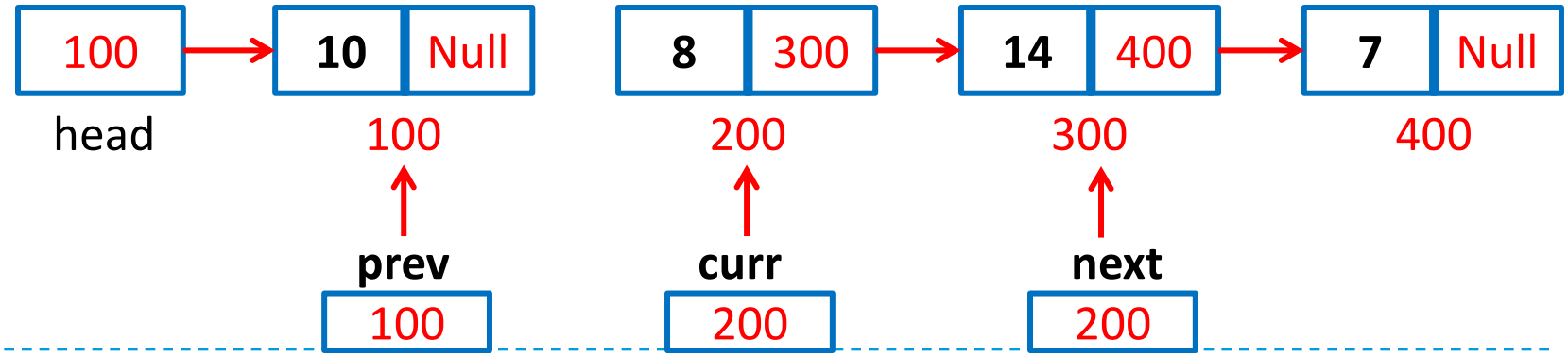
Membalik List secara Iteratif

- Diperlukan tiga 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



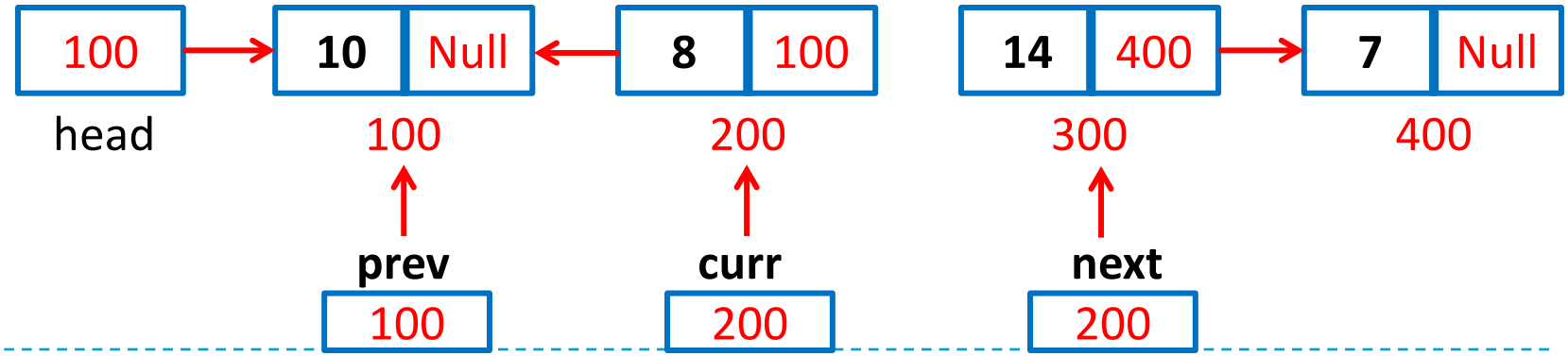
Membalik List secara Iteratif

- Diperlukan tiga 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



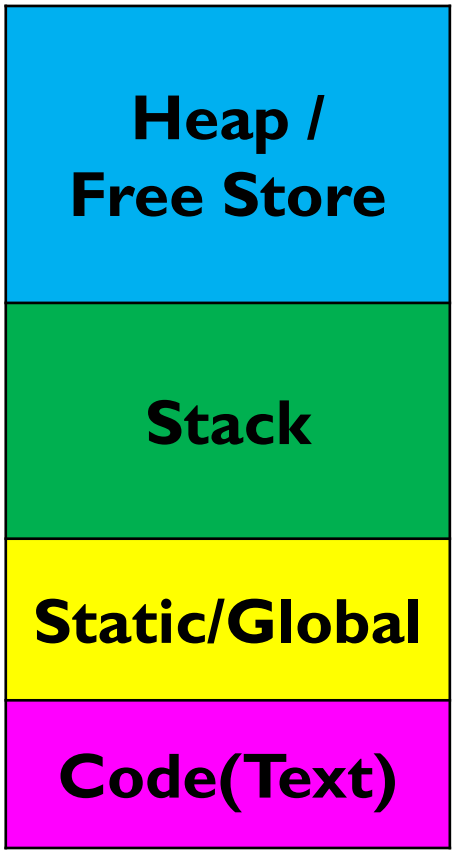
Membalik List secara Iteratif

- **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

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

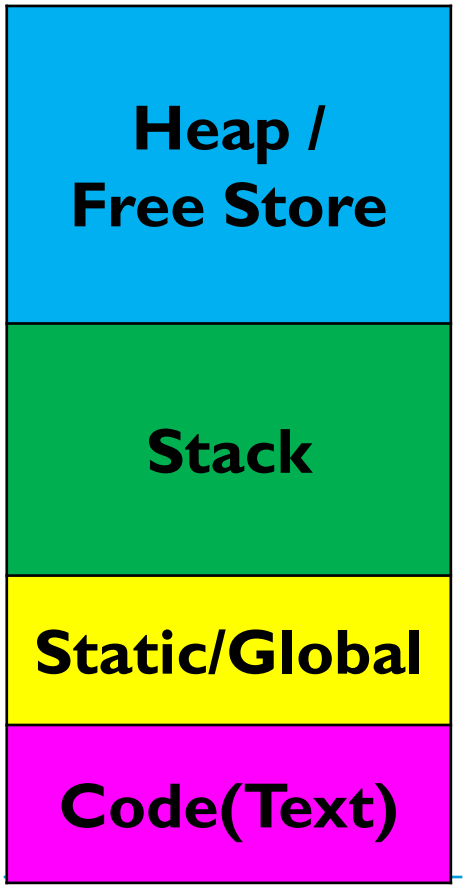


Bersifat **Dinamis**:
Ukuran memori dapat berubah ketika program dijalankan

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

Fungsi Rekursif pada Memory

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



Memori yang dapat di-*request* selama program berjalan

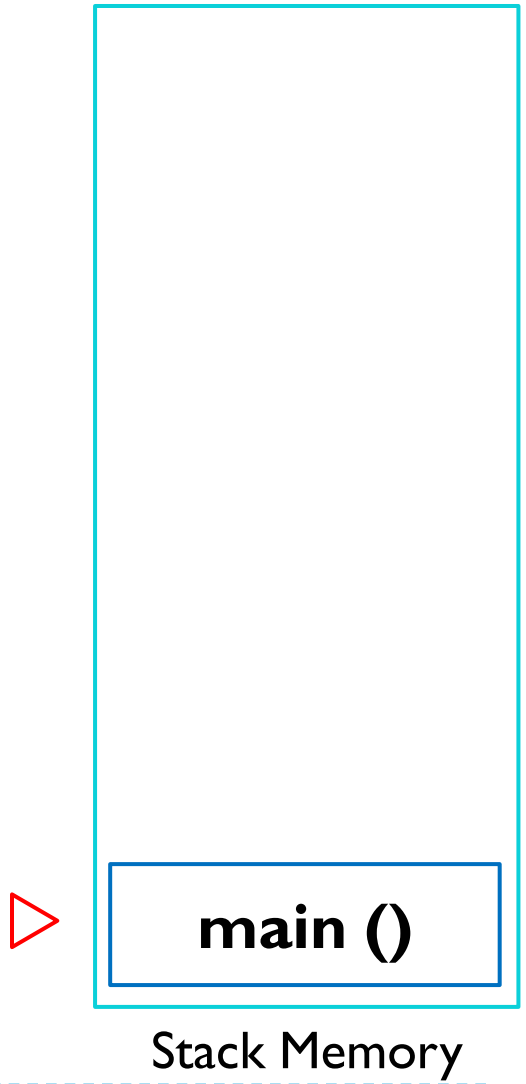
Menyimpan semua informasi tentang pemanggilan **fungsi** untuk menyimpan semua variabel lokal

Menyimpan variabel global selama program berjalan

Menyimpan semua perintah yang akan dijalankan/dieksekusi

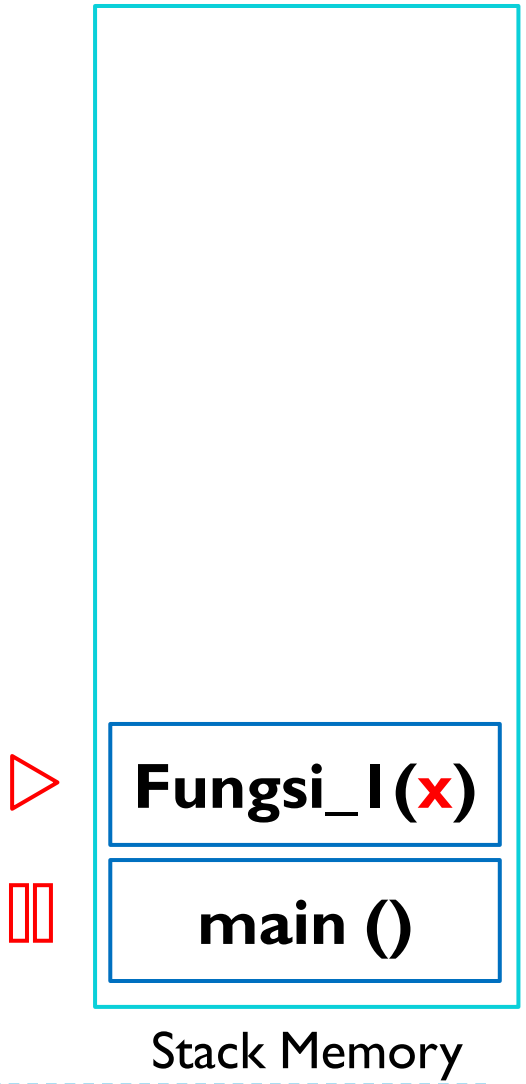
Fungsi Rekursif pada Memory

- Stack memory dialokasikan ketika terjadi pemanggilan fungsi tertentu
- 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



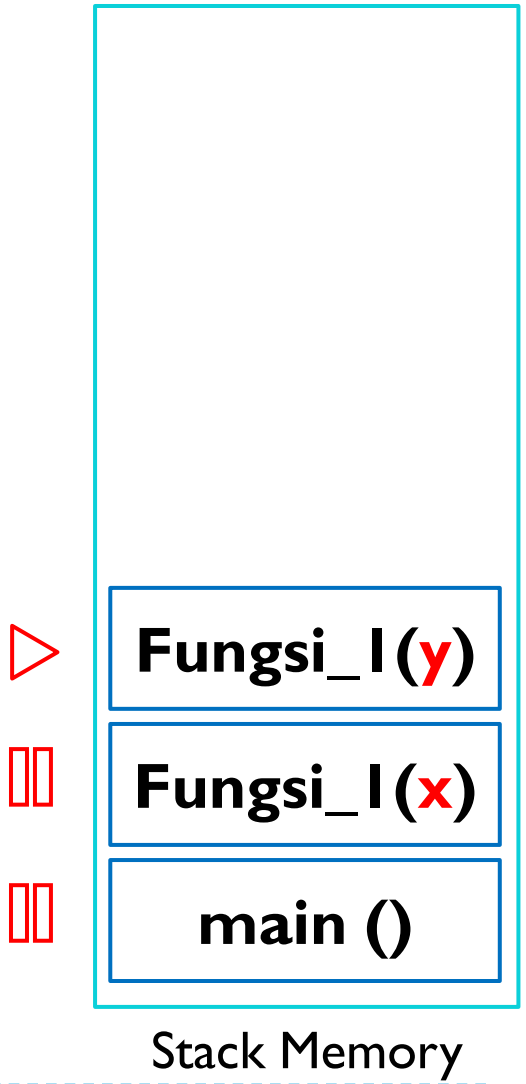
Fungsi Rekursif pada Memory

- 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



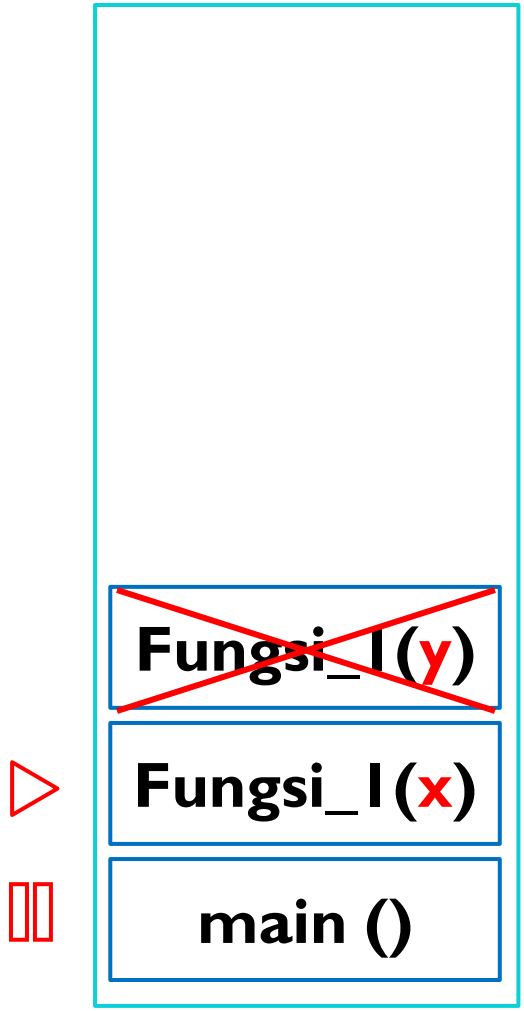
Fungsi Rekursif pada Memory

- 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

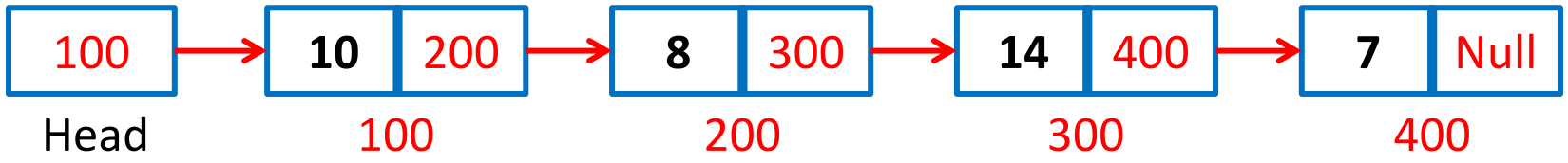
- Stack memory dialokasikan ketika terjadi pemanggilan fungsi tertentu
- Nilai dan kondisi terakhir pada pemanggilan fungsi tetap tersimpan pada stack memory
- Fungsi baru akan ditumpuk (berjalan) diatas fungsi sebelumnya (sekaligus menghentikan sementara eksekusi fungsi sebelumnya) pada stack memory
- Jika fungsi pada tumpukan paling atas selesai dijalankan, fungsi pada tumpukan dibawahnya akan dijalankan kembali



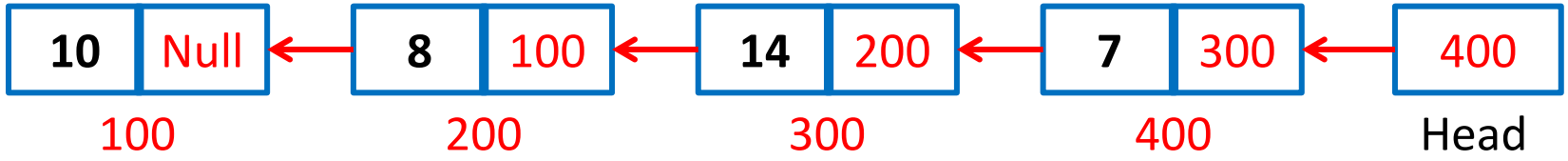
Stack Memory

Membalik List secara Rekursif

- Input



- Output

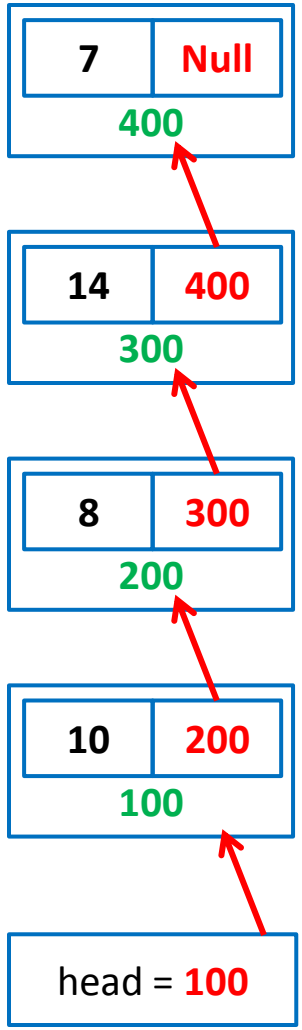


Membalik List secara Rekursif

- **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;
}
```

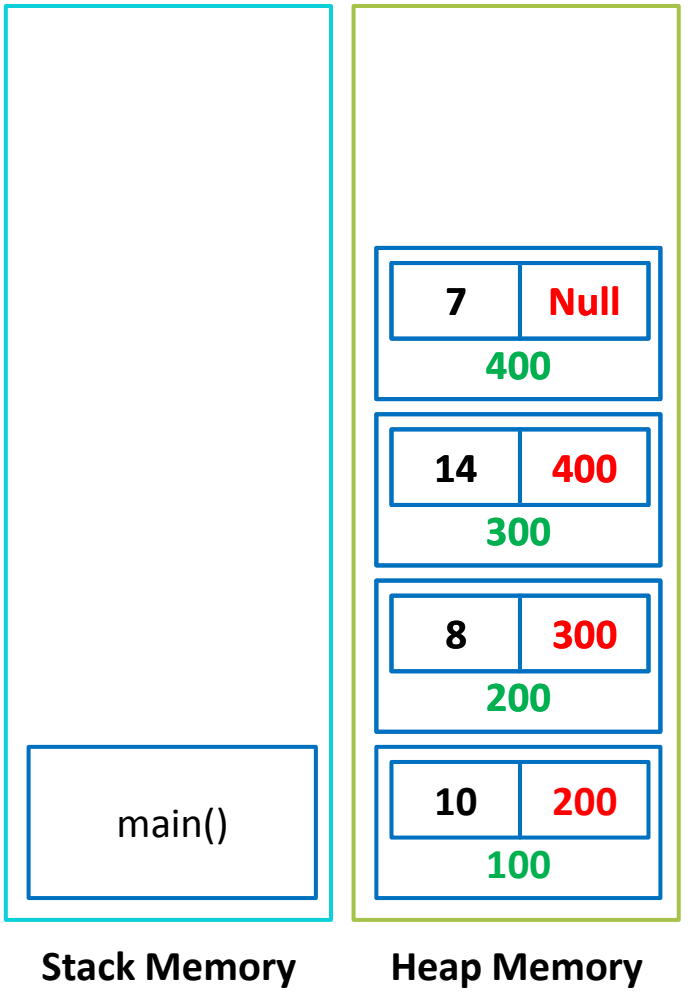


Membalik List secara Rekursif

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
}
```

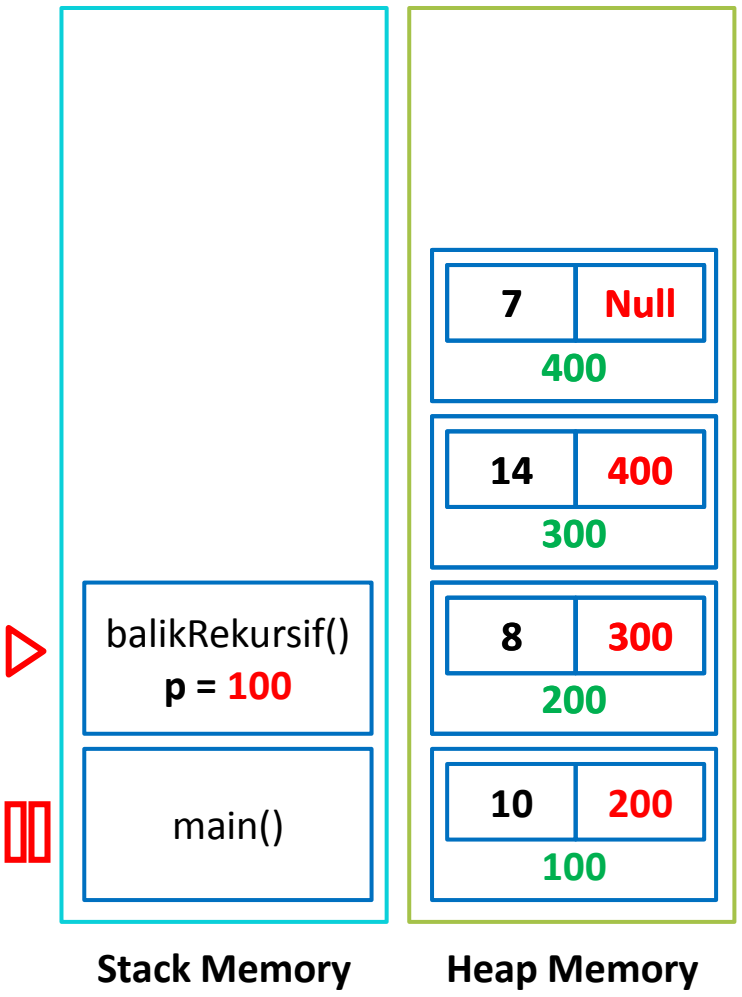


Membalik List secara Rekursif

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
}
```

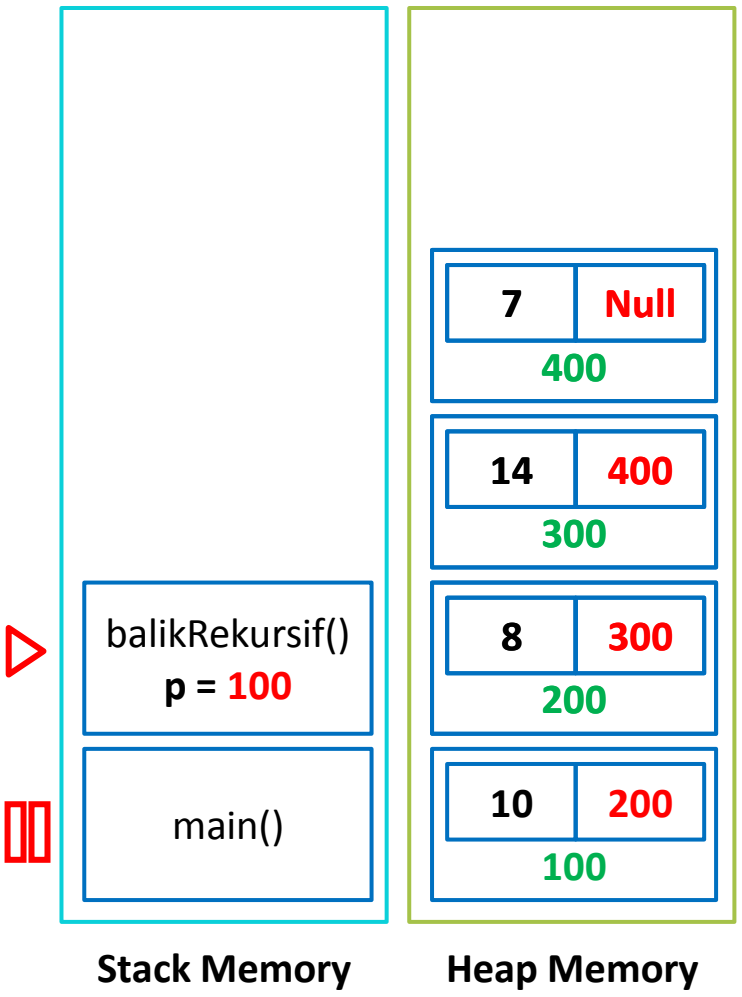


Membalik List secara Rekursif

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
}
```

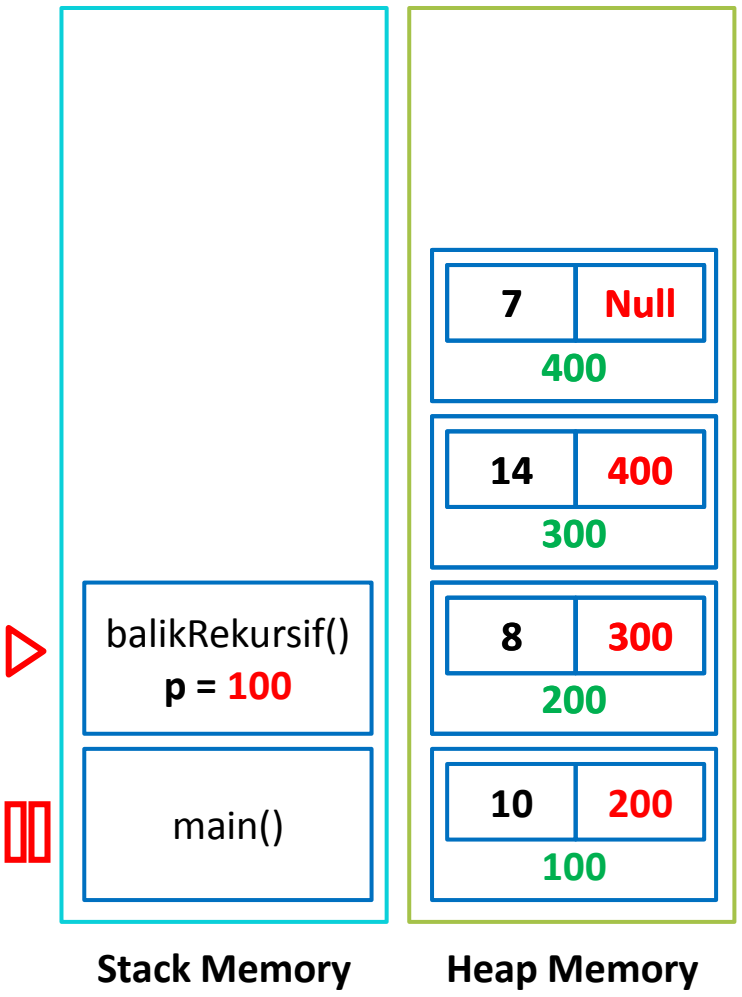


Membalik List secara Rekursif

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
}
```

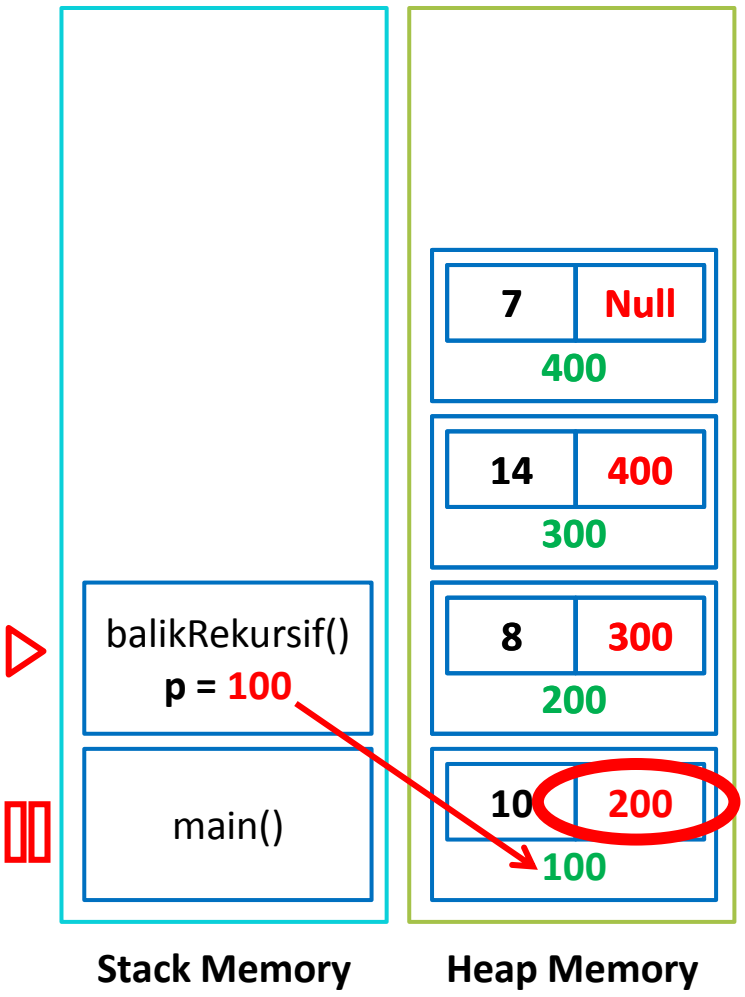


Membalik List secara Rekursif

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
}
```

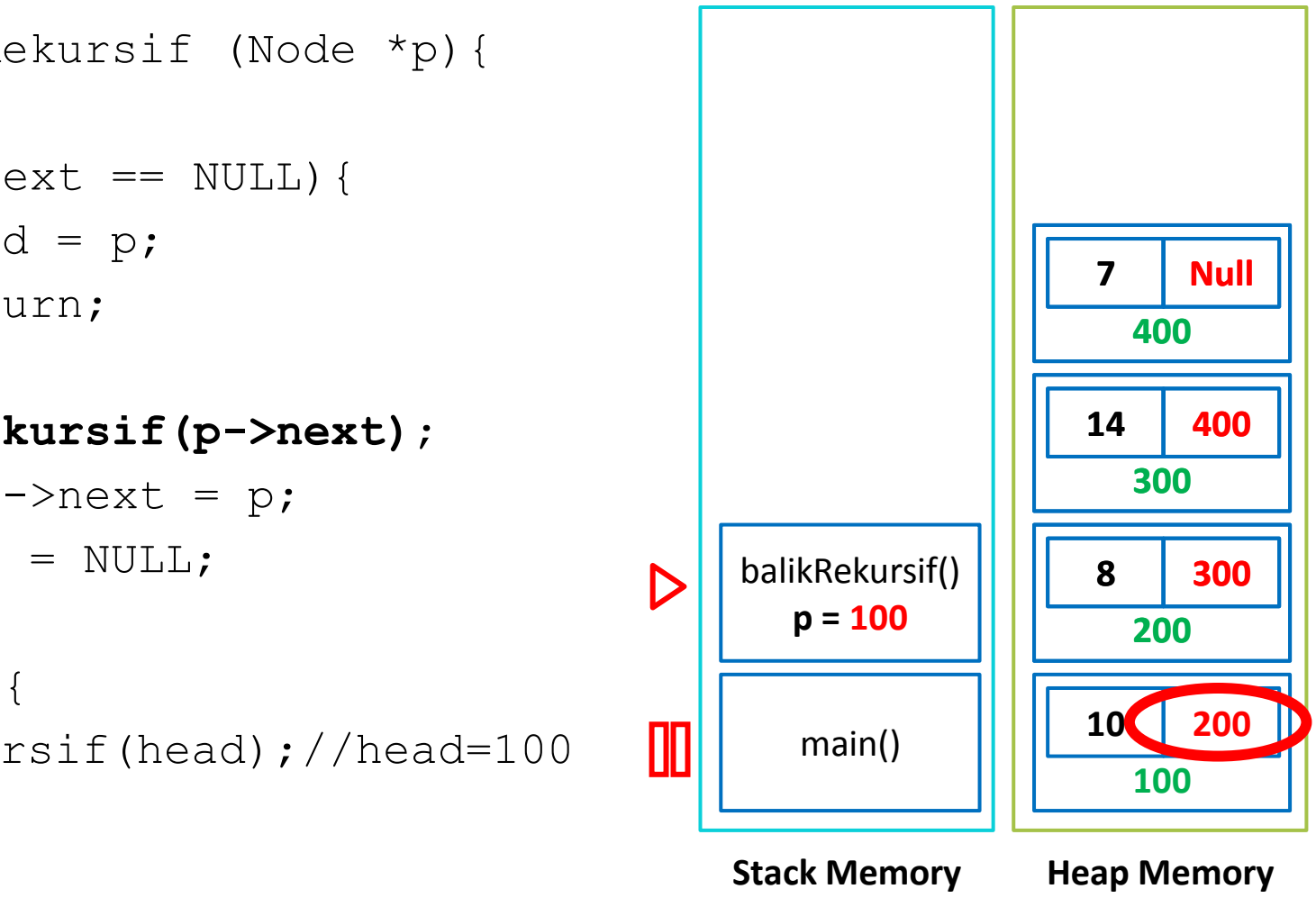


Membalik List secara Rekursif

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
}
```

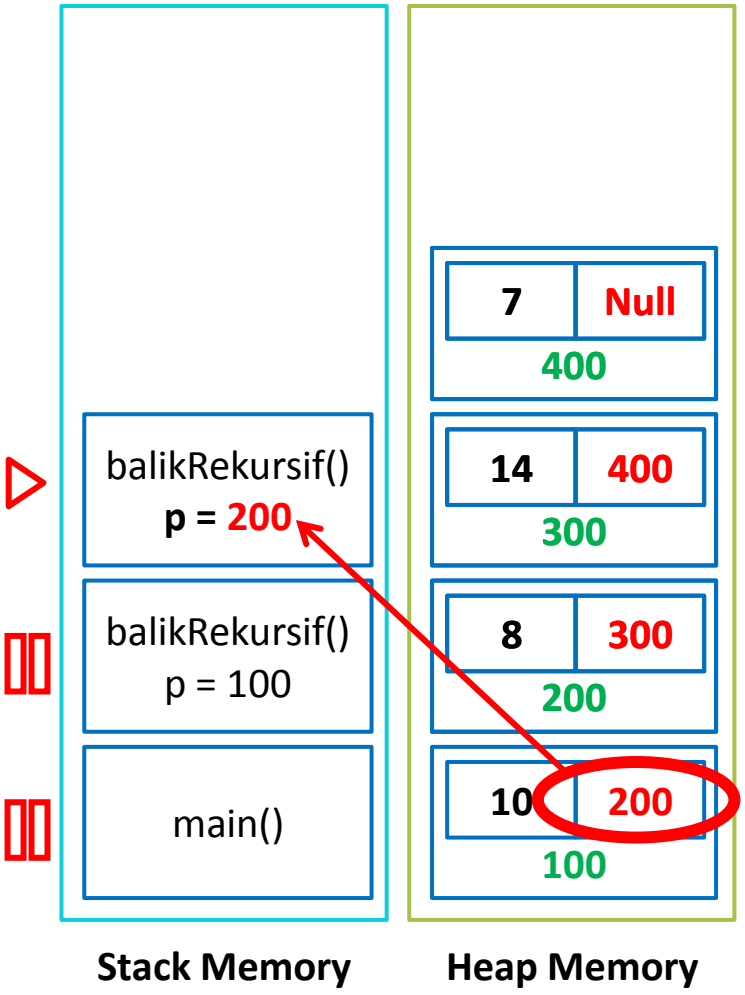


Membalik List secara Rekursif

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
}
```

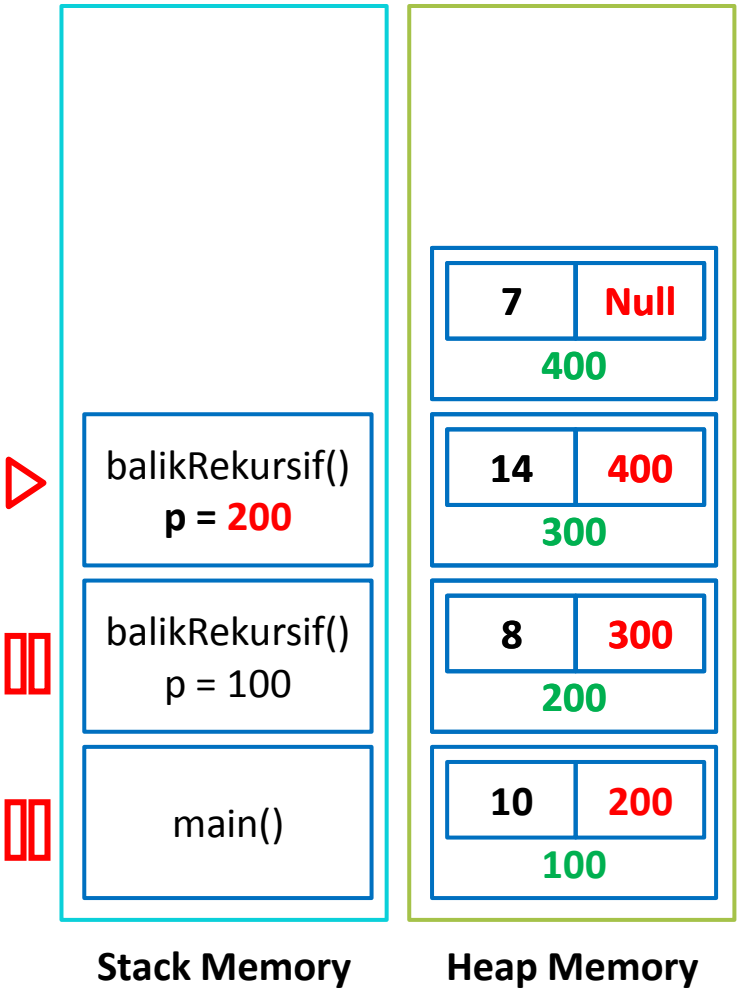


Membalik List secara Rekursif

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
}
```

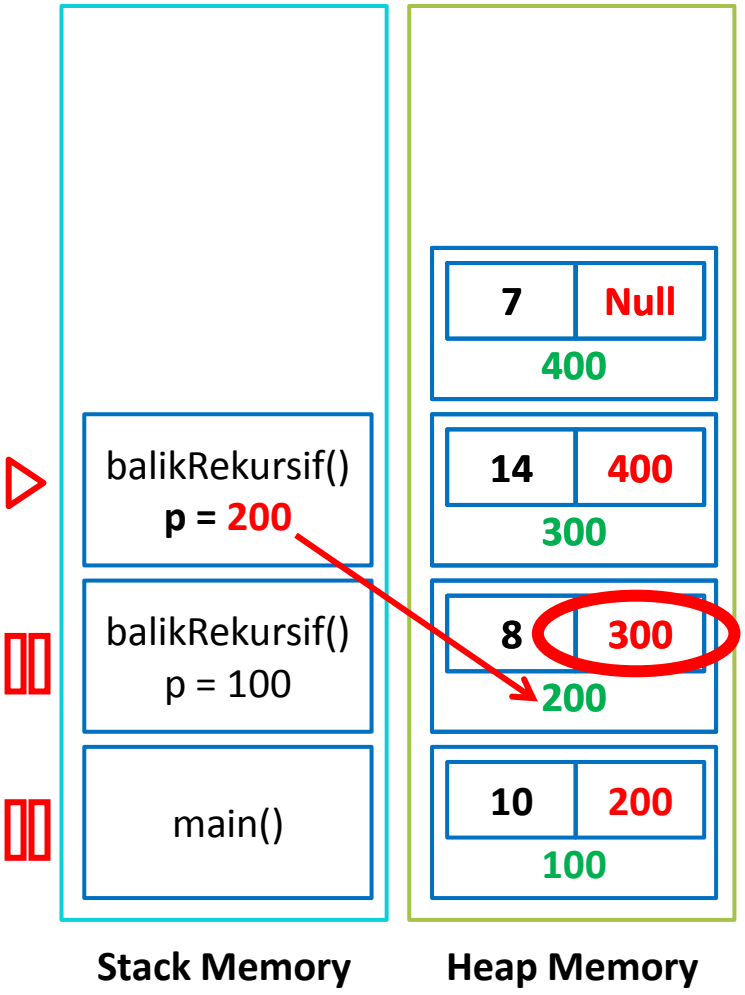


Membalik List secara Rekursif

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
}
```

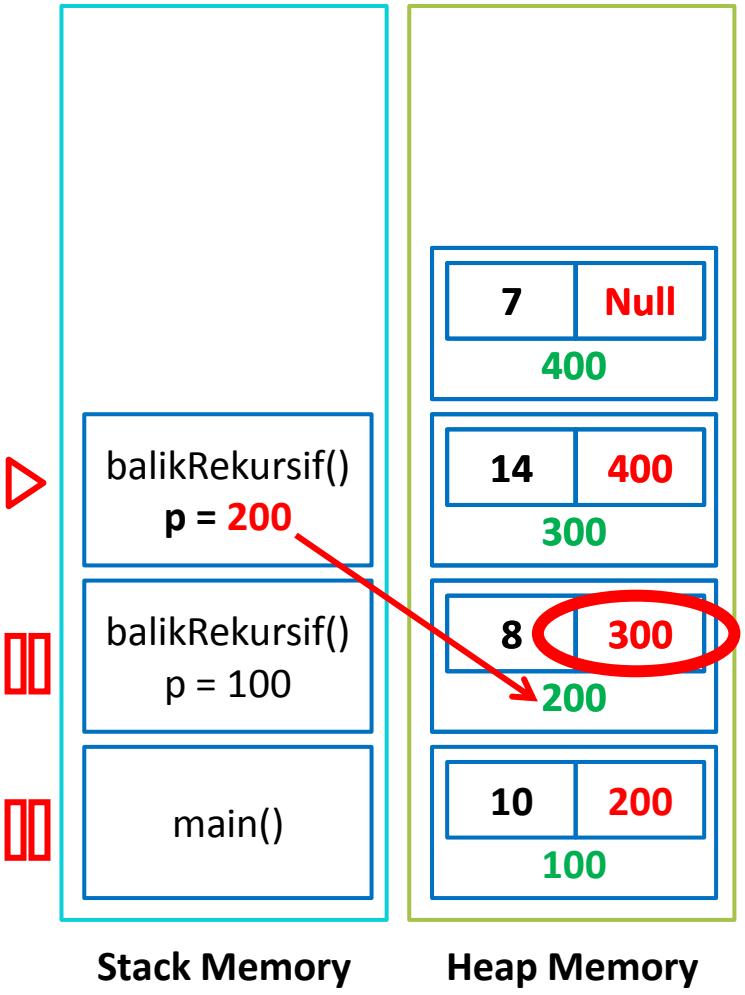


Membalik List secara Rekursif

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
}
```

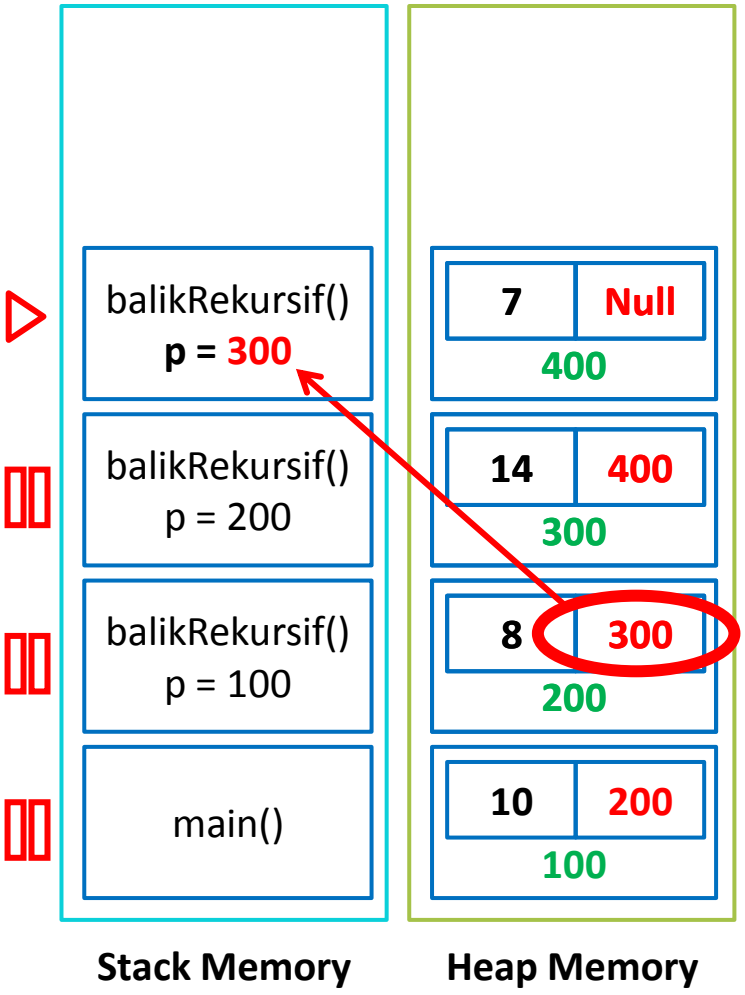


Membalik List secara Rekursif

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
}
```

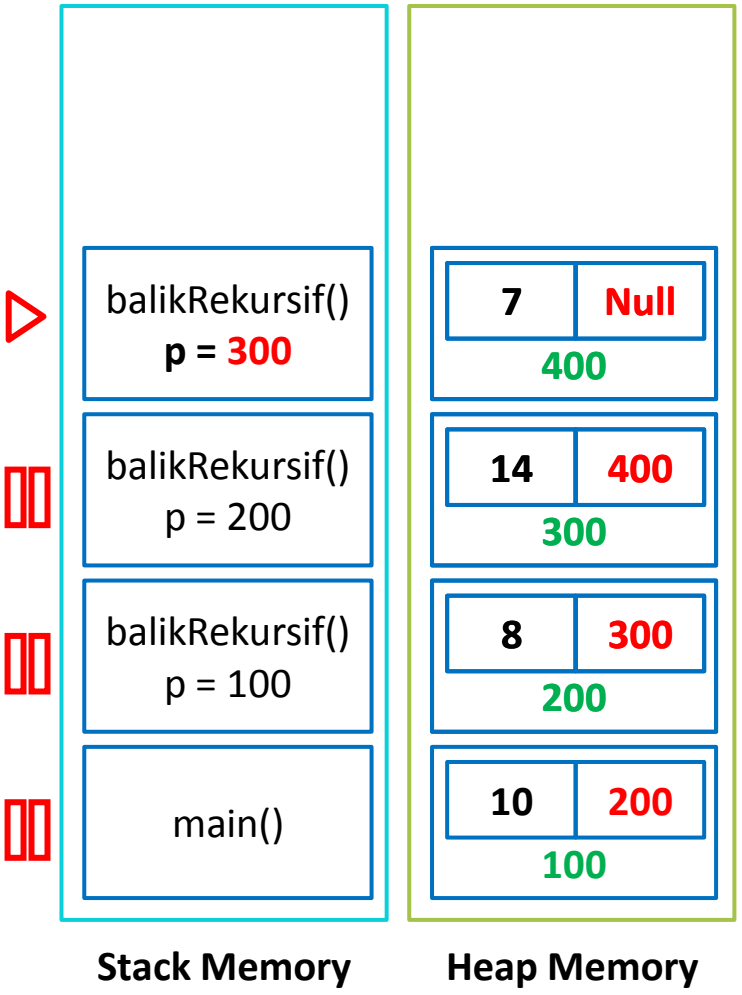


Membalik List secara Rekursif

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
}
```

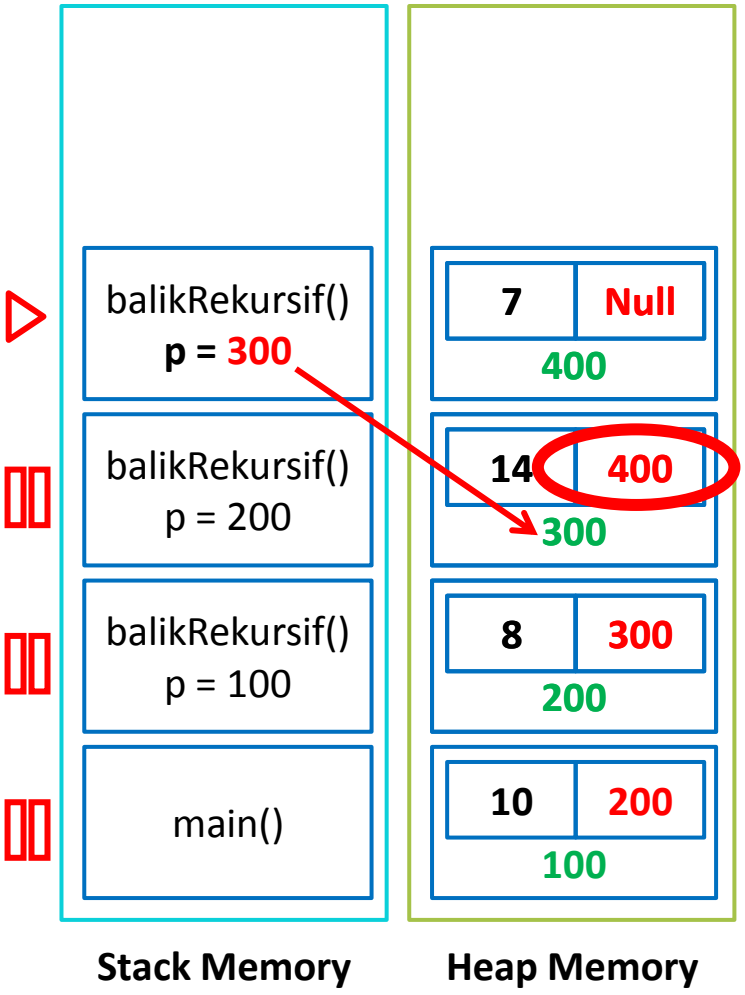


Membalik List secara Rekursif

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
}
```

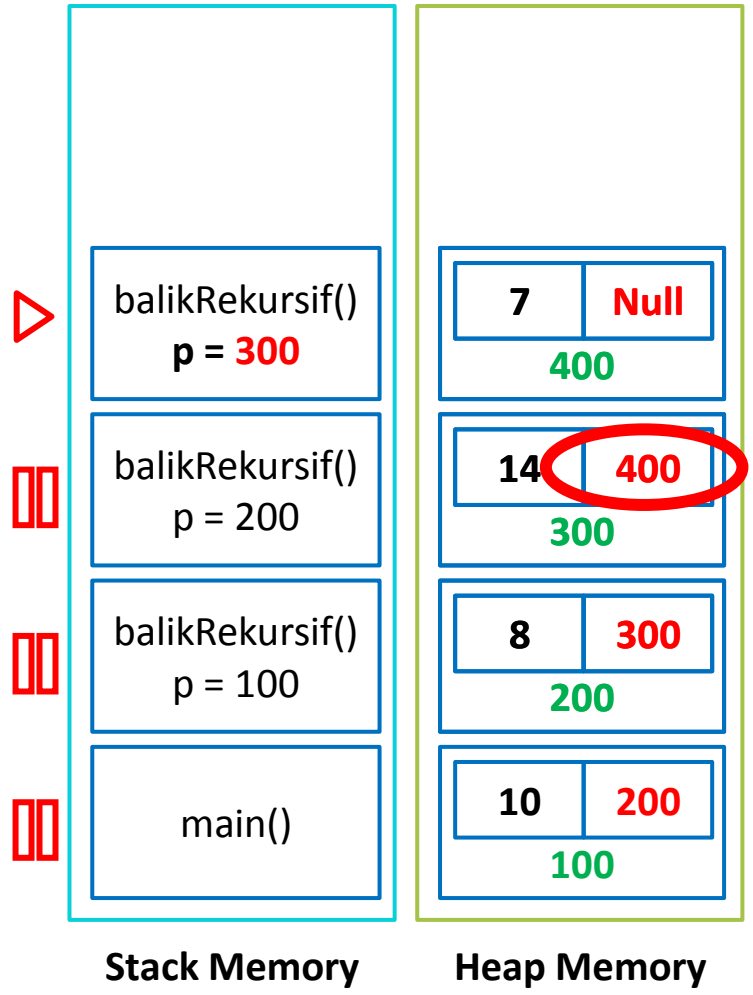


Membalik List secara Rekursif

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
}
```

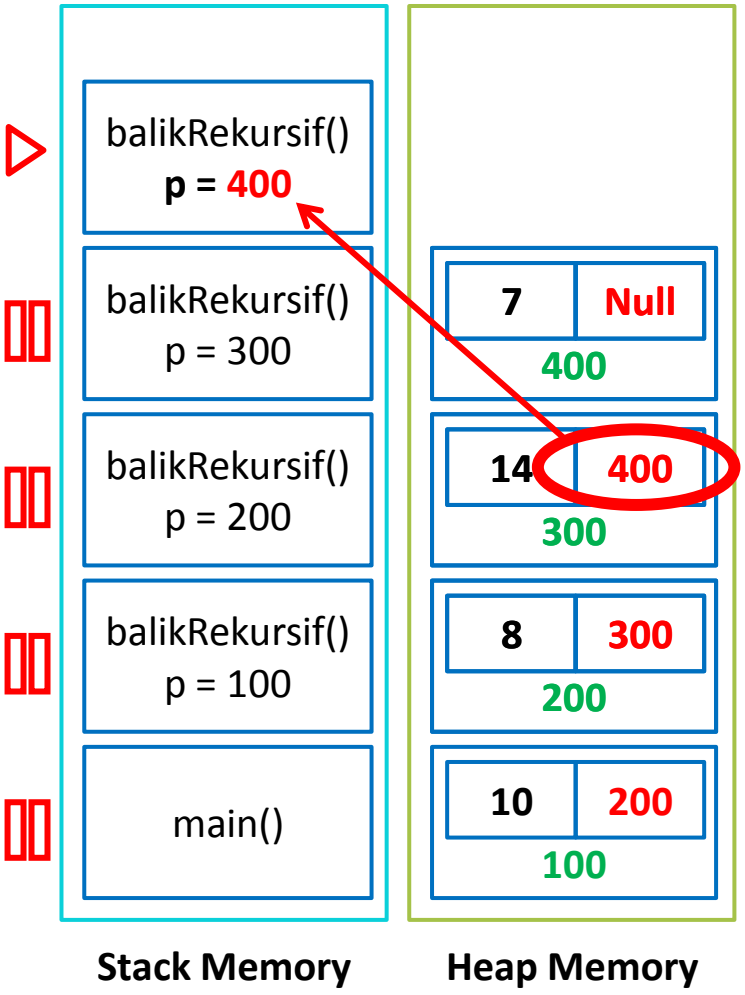


Membalik List secara Rekursif

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
}
```

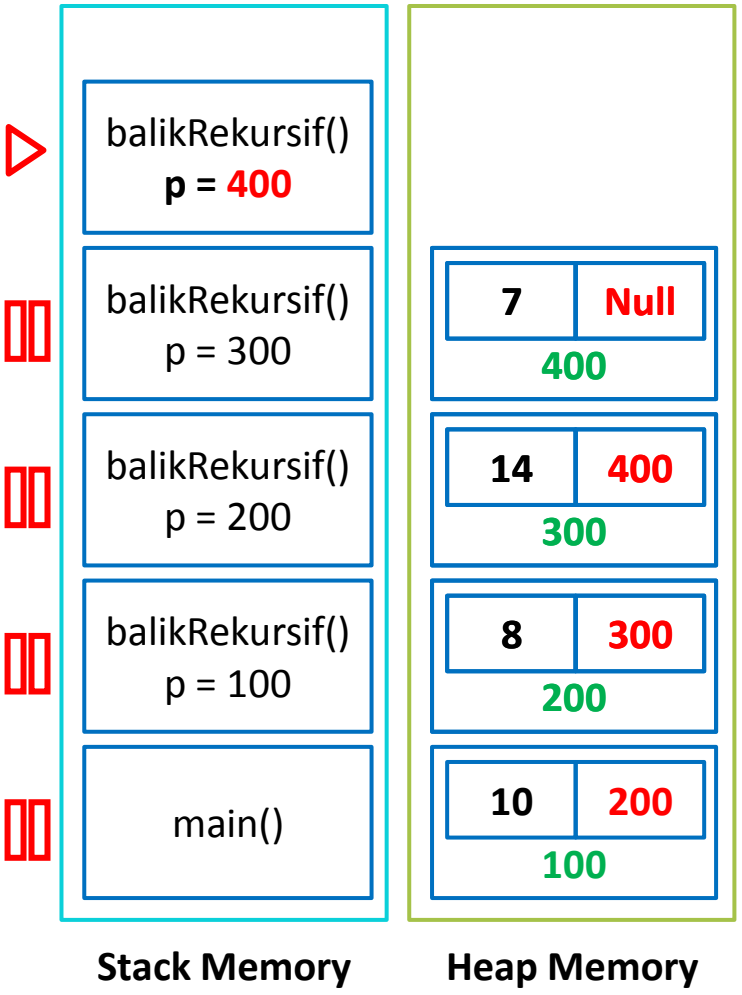


Membalik List secara Rekursif

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
}
```

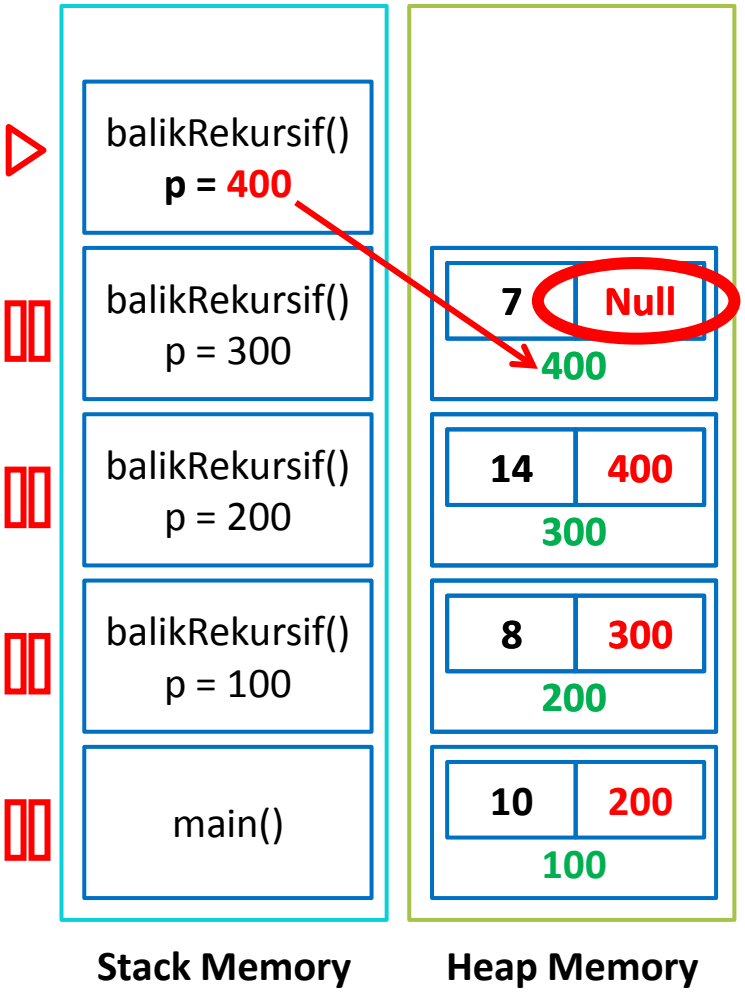


Membalik List secara Rekursif

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
}
```

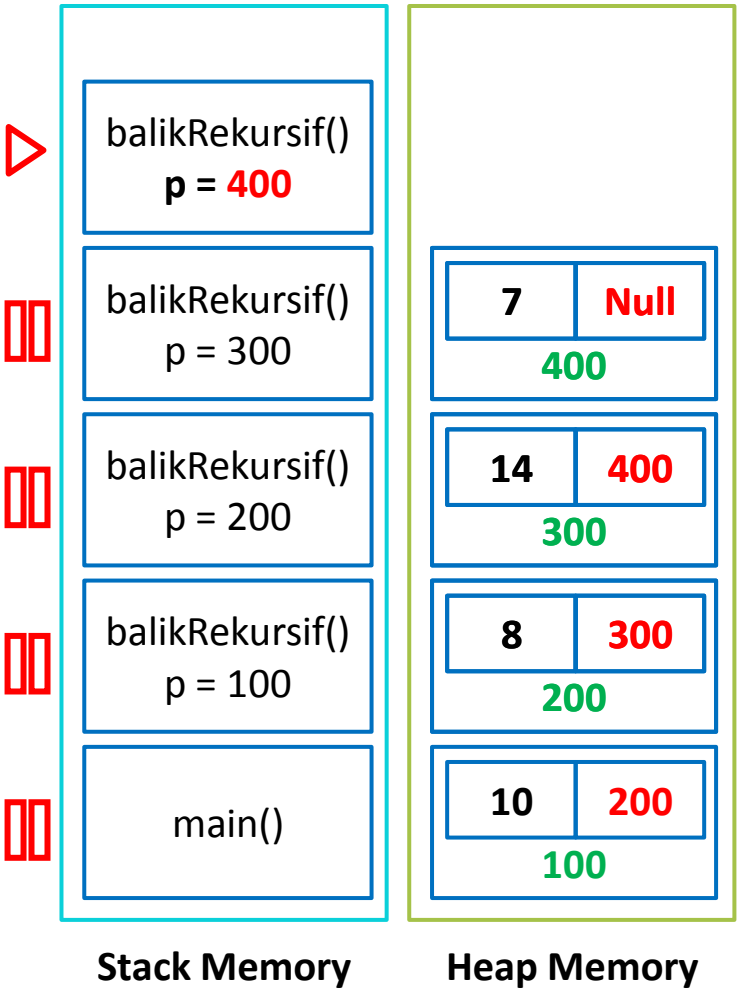


Membalik List secara Rekursif

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
}
```

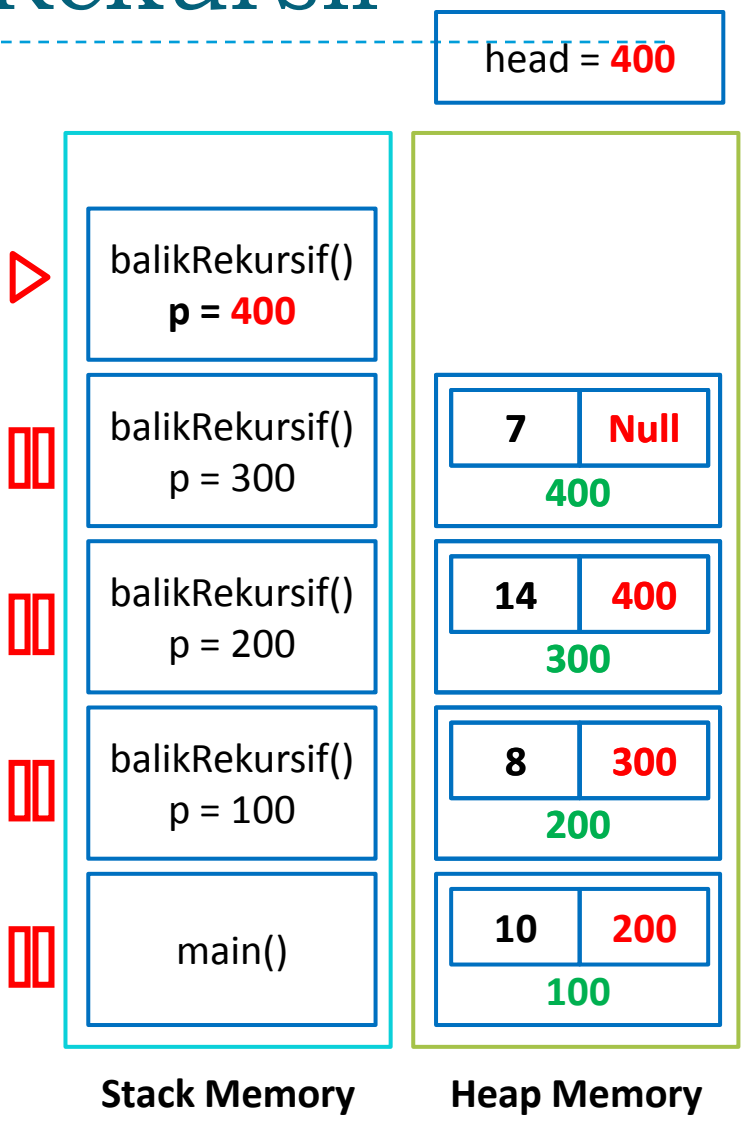


Membalik List secara Rekursif

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
}
```

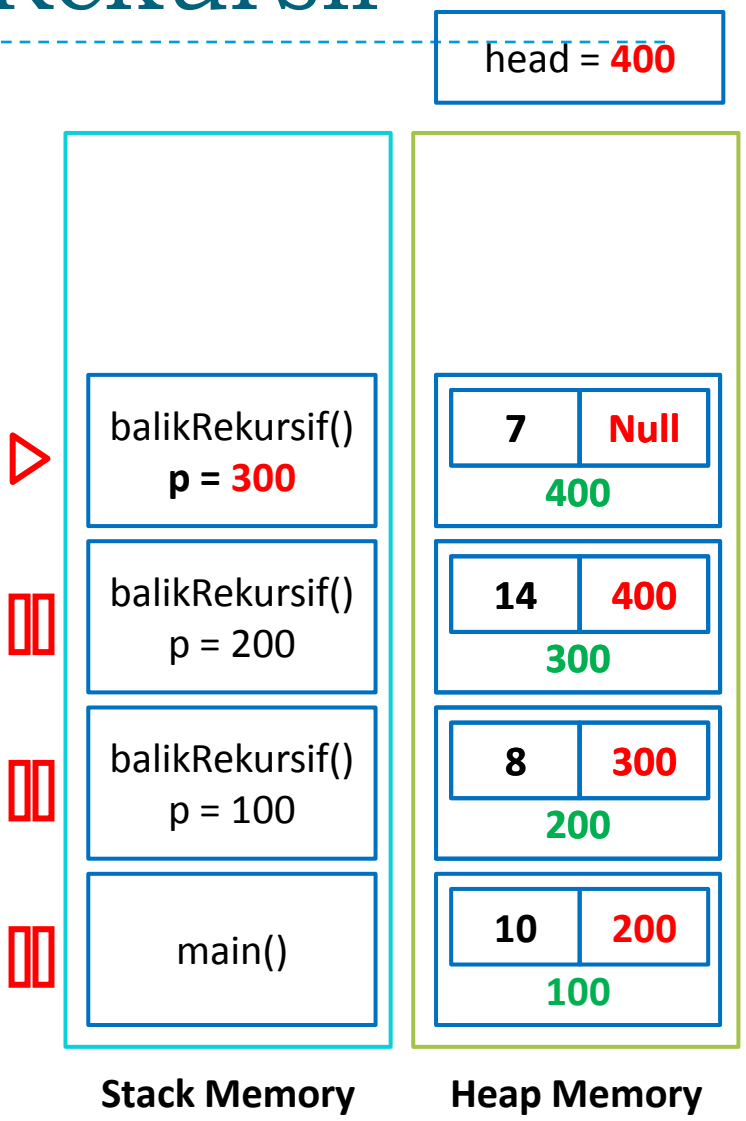


Membalik List secara Rekursif

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
}
```

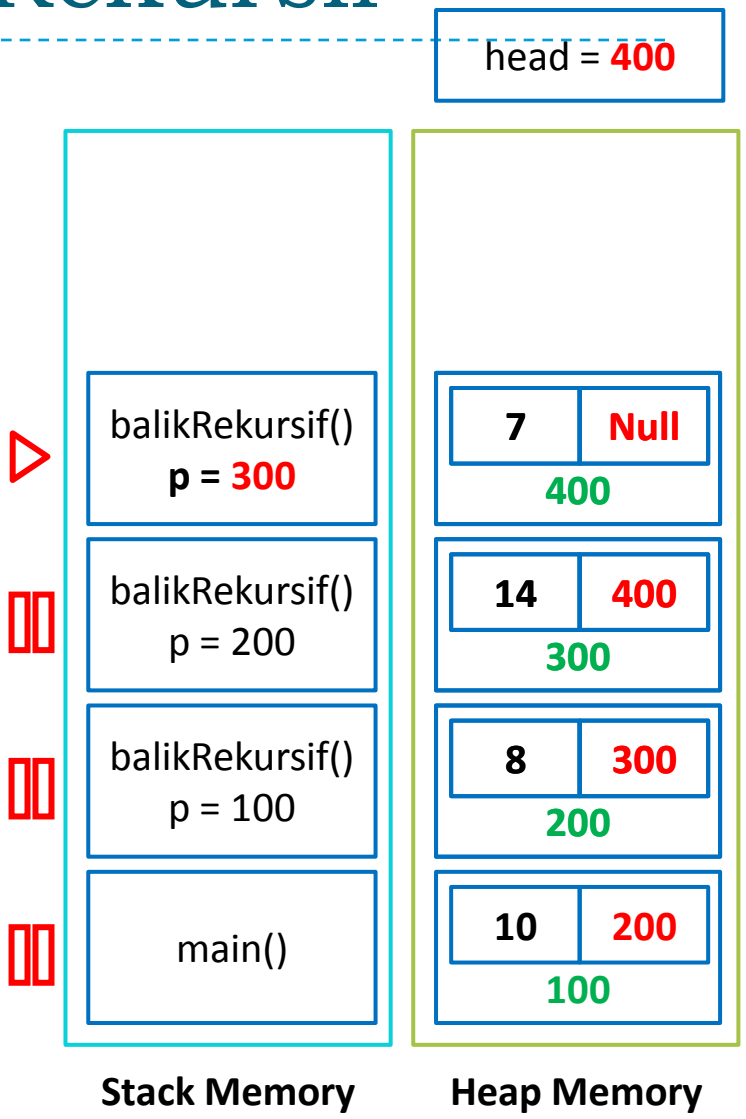


Membalik List secara Rekursif

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
}
```

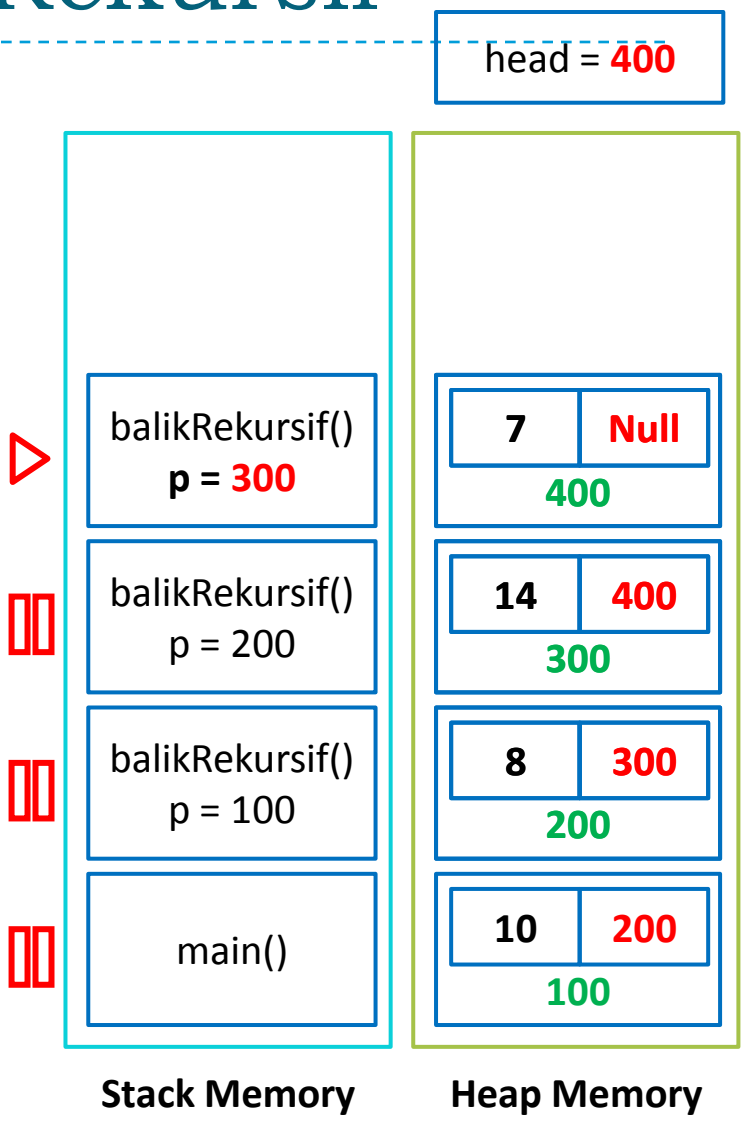


Membalik List secara Rekursif

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

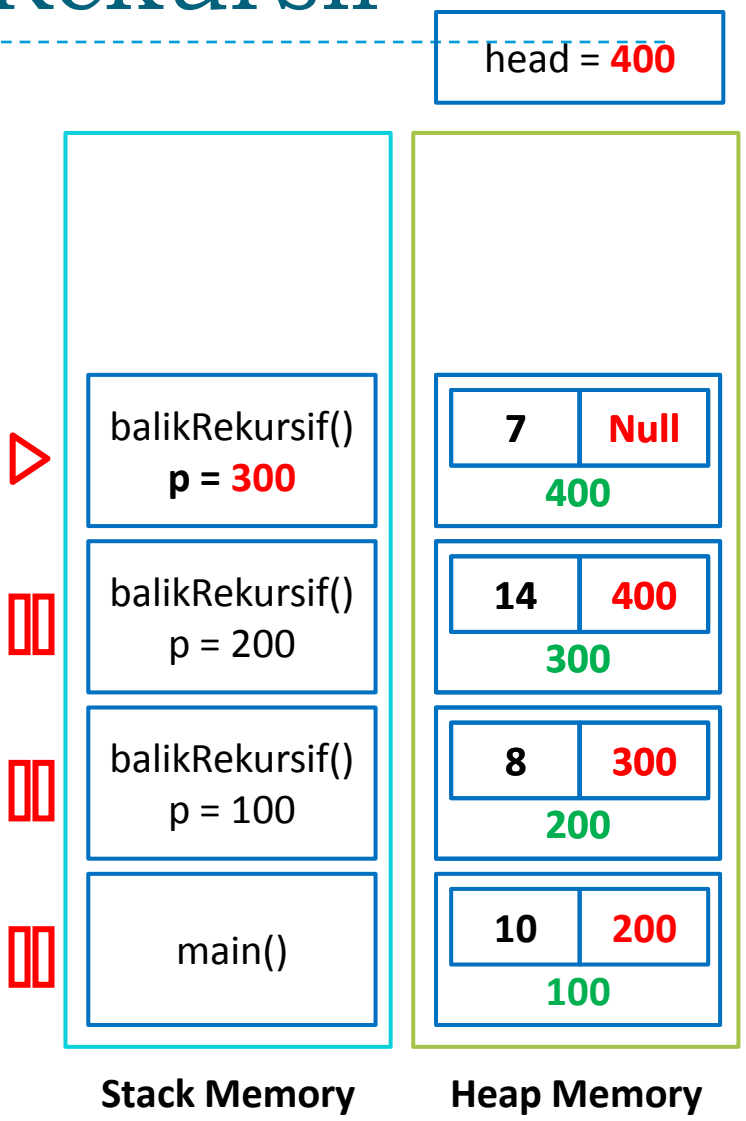


Membalik List secara Rekursif

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

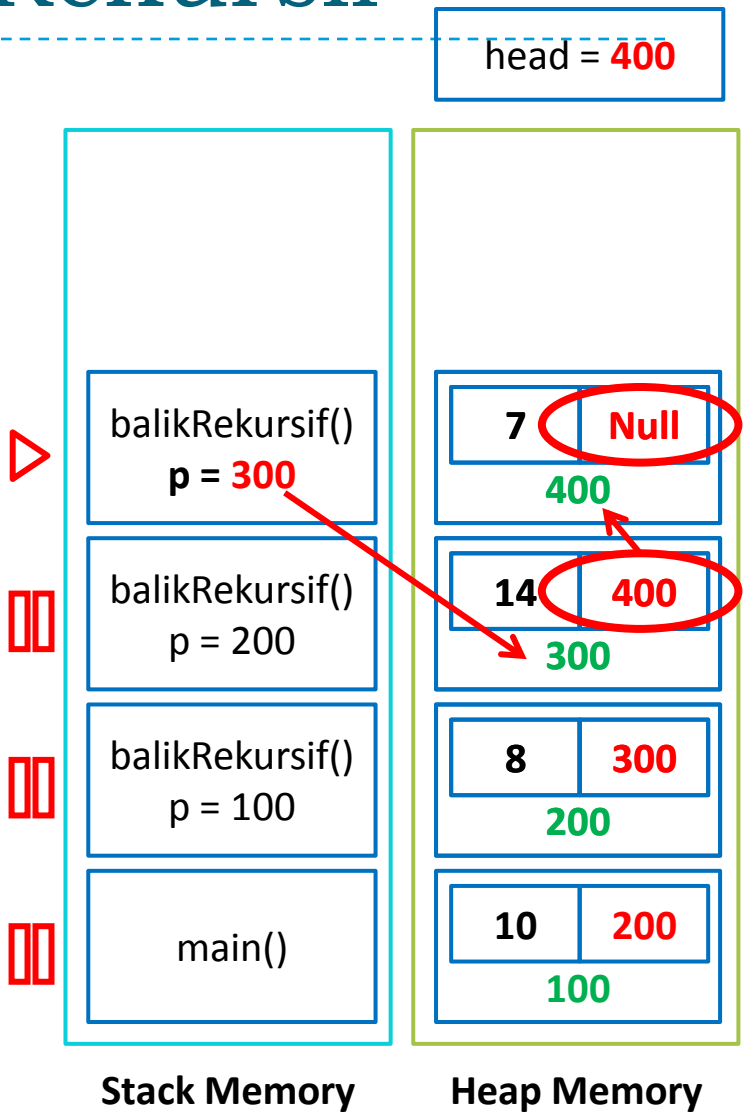


Membalik List secara Rekursif

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

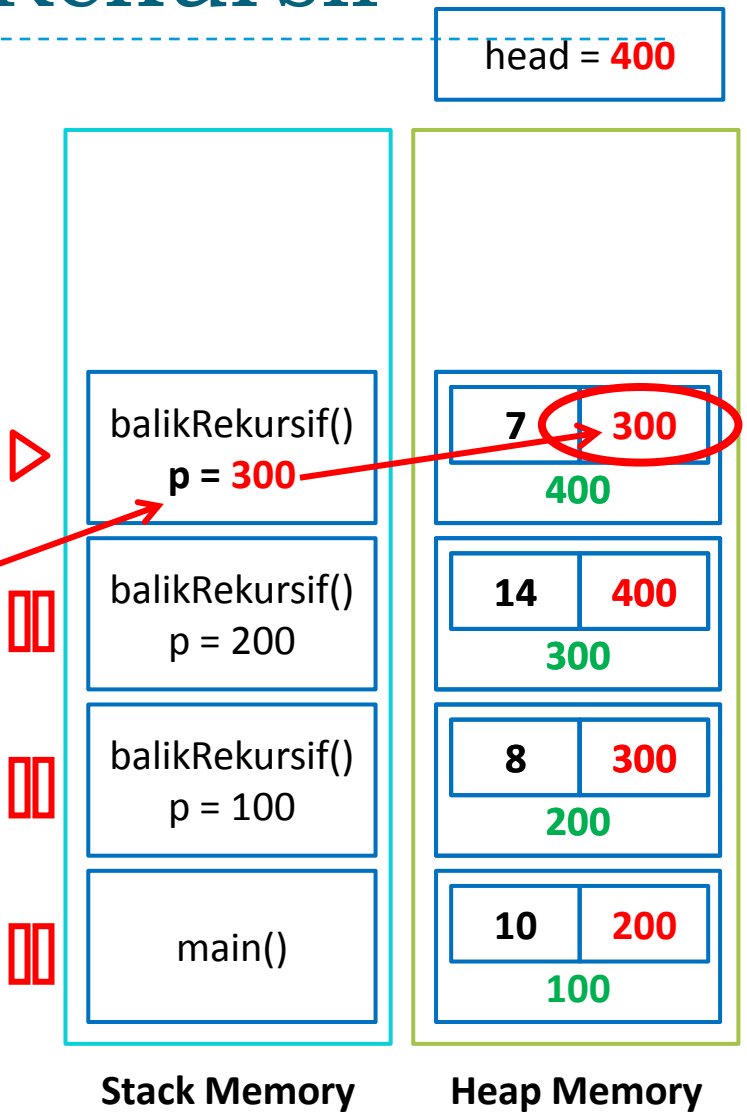


Membalik List secara Rekursif

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

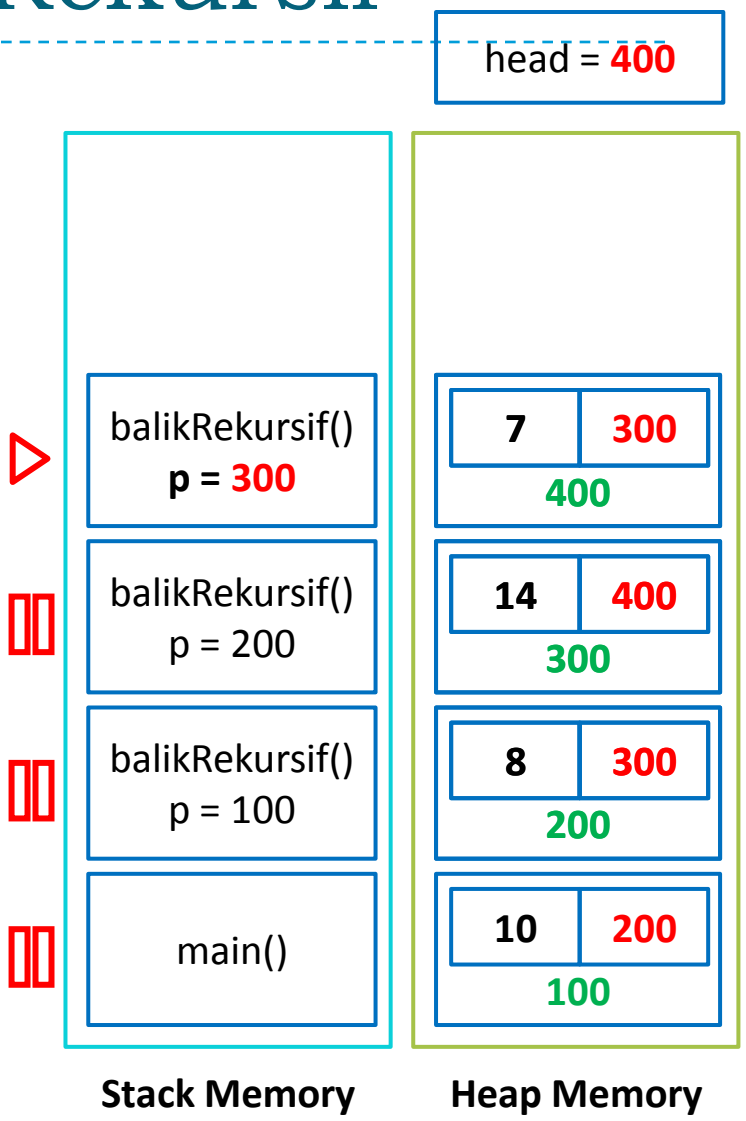


Membalik List secara Rekursif

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

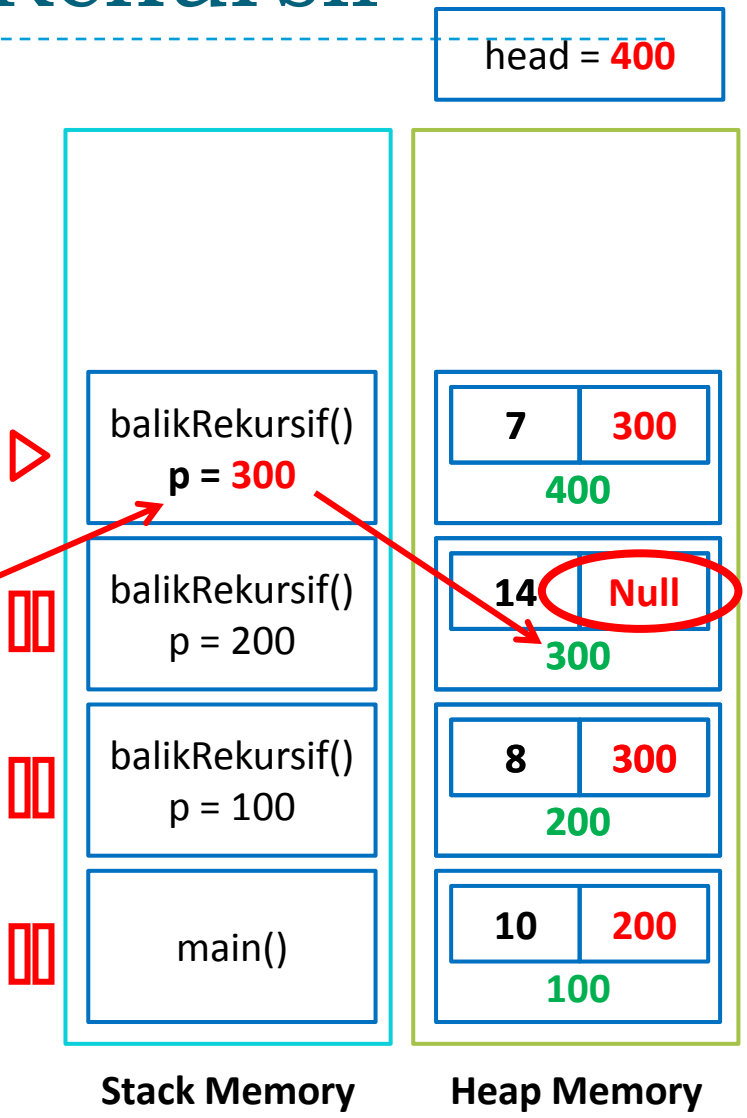


Membalik List secara Rekursif

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

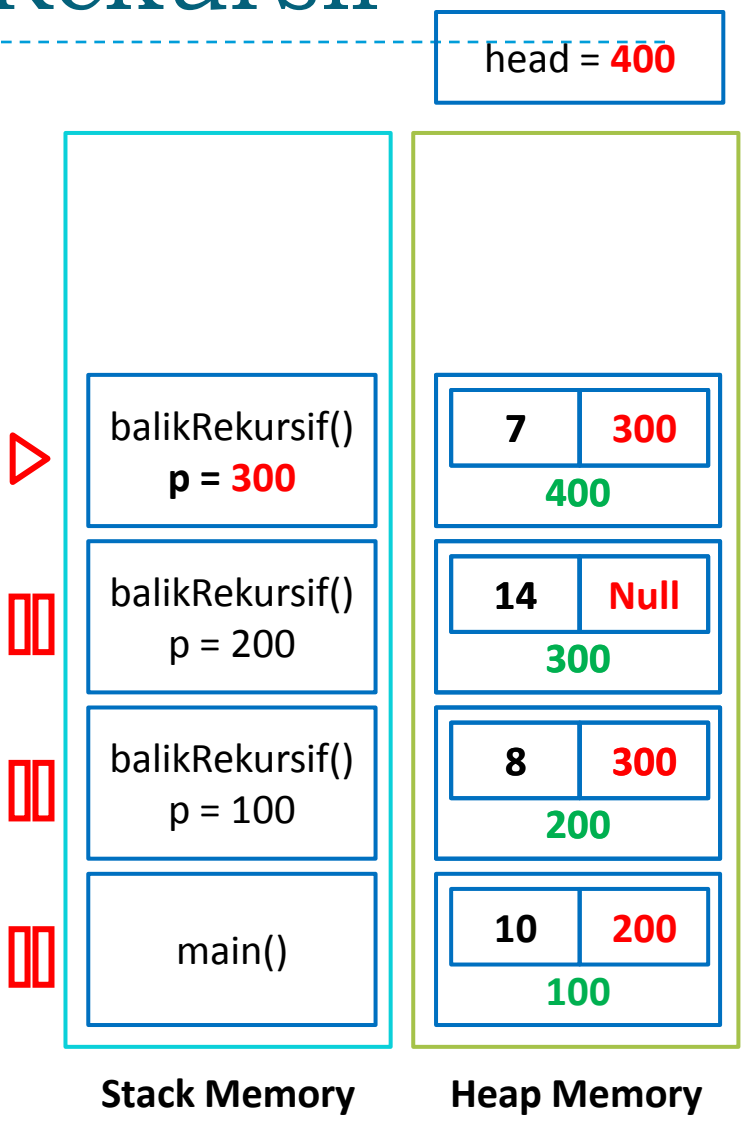


Membalik List secara Rekursif

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
}
```

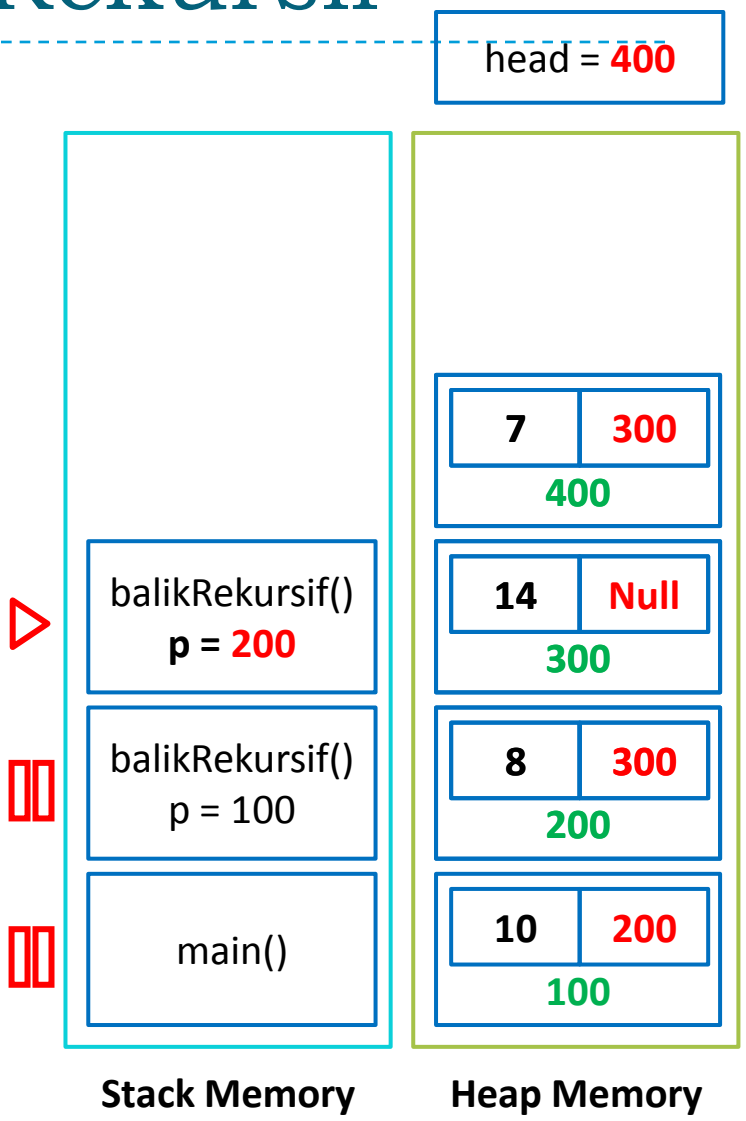


Membalik List secara Rekursif

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
}
```

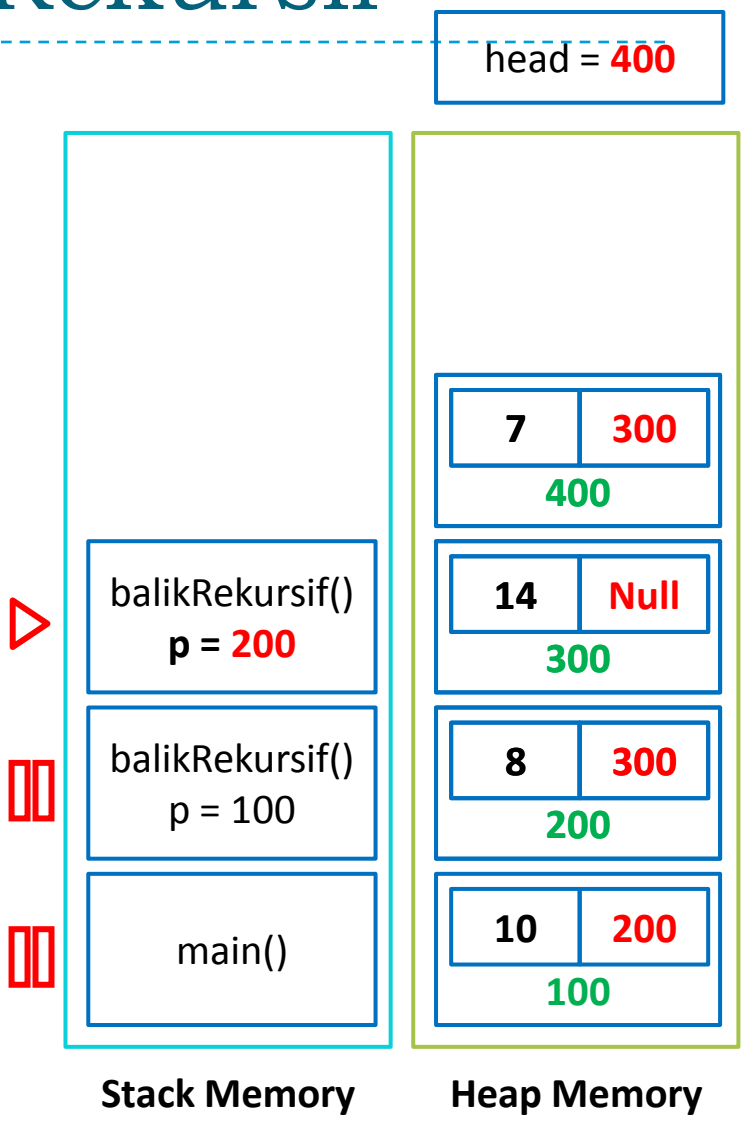


Membalik List secara Rekursif

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
}
```

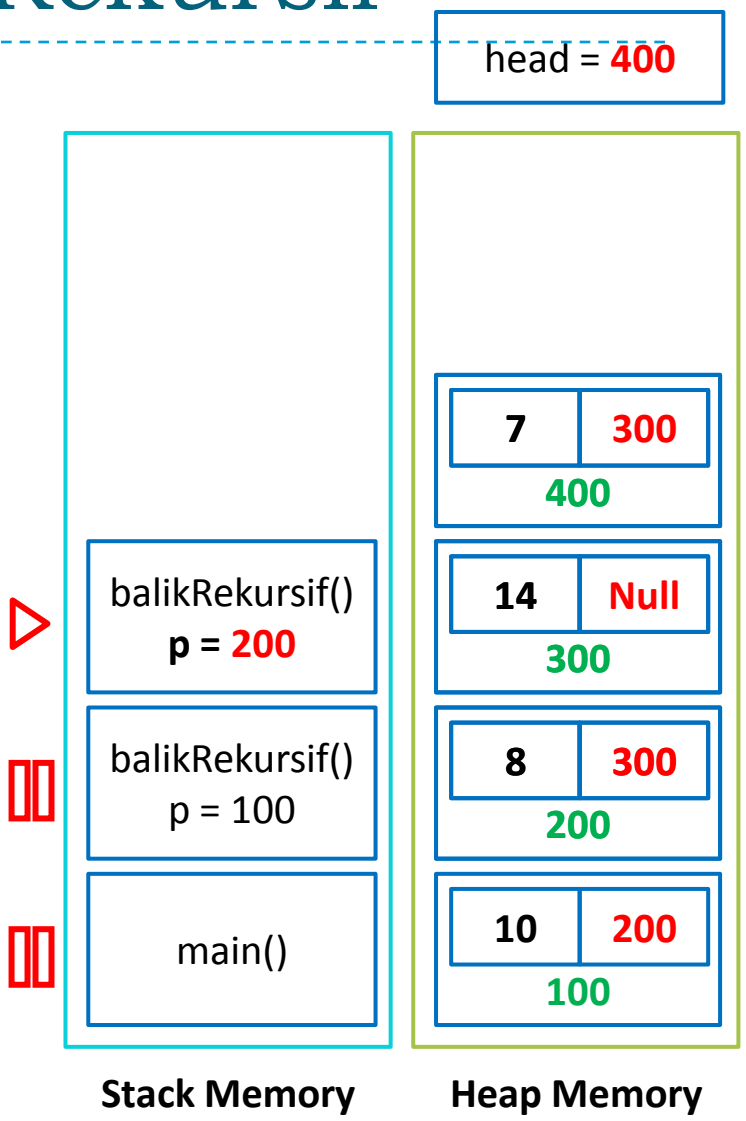


Membalik List secara Rekursif

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

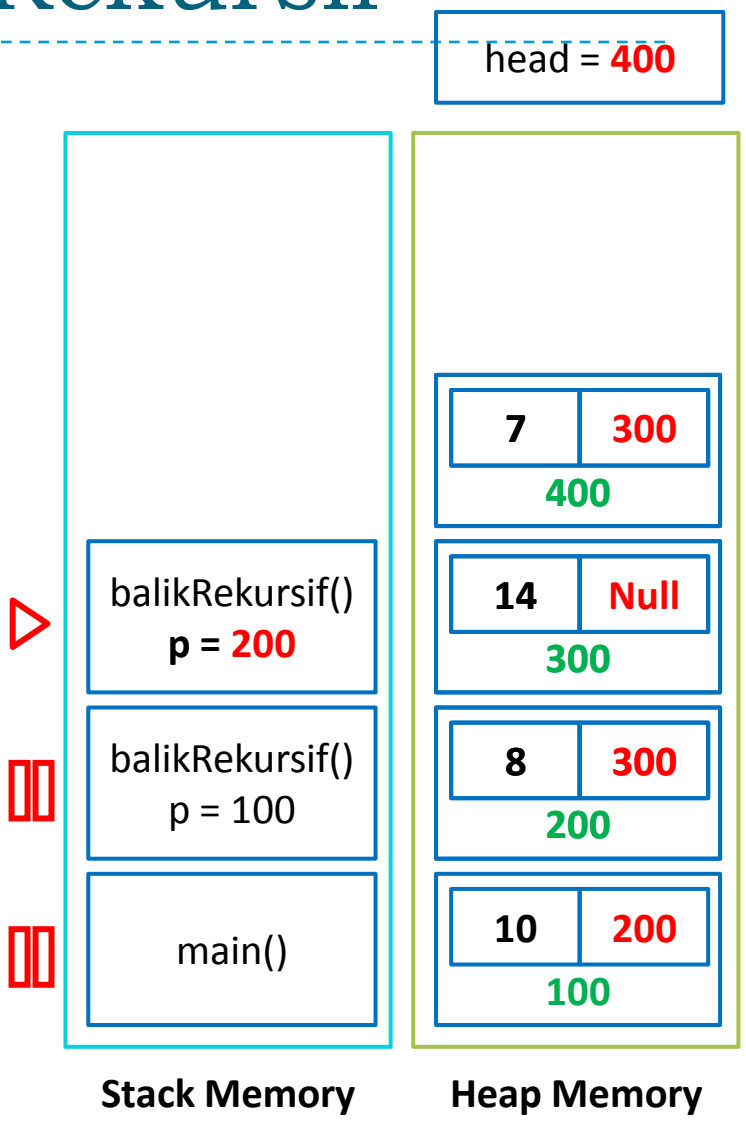


Membalik List secara Rekursif

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

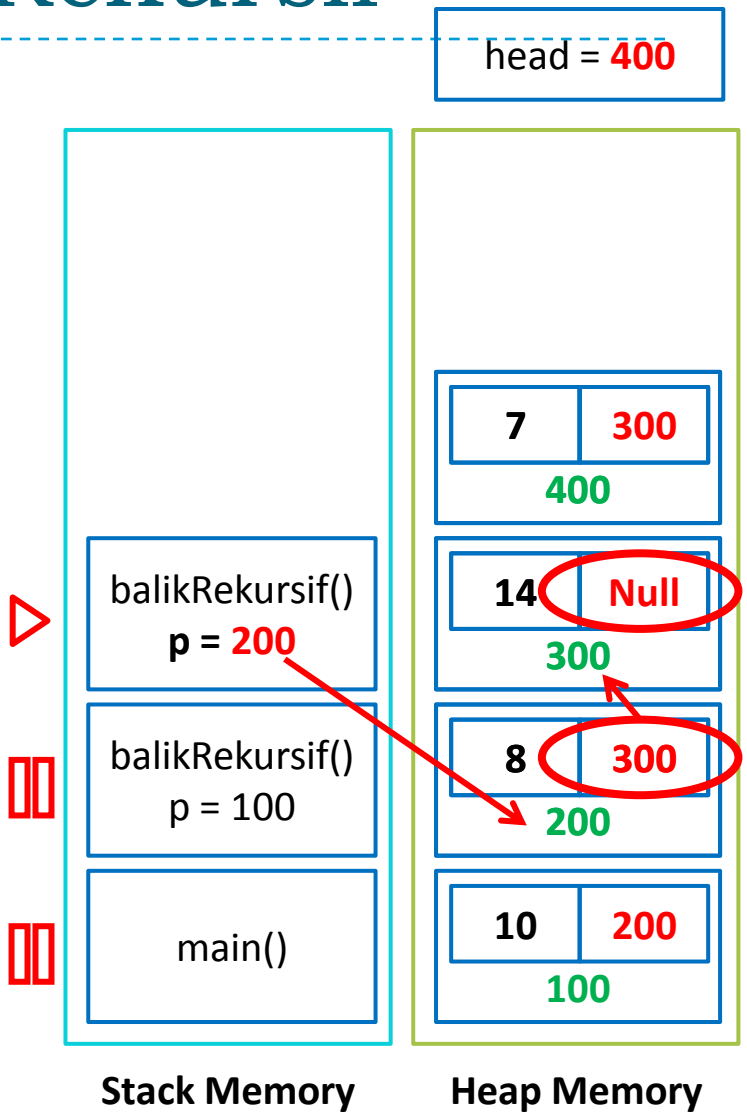


Membalik List secara Rekursif

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

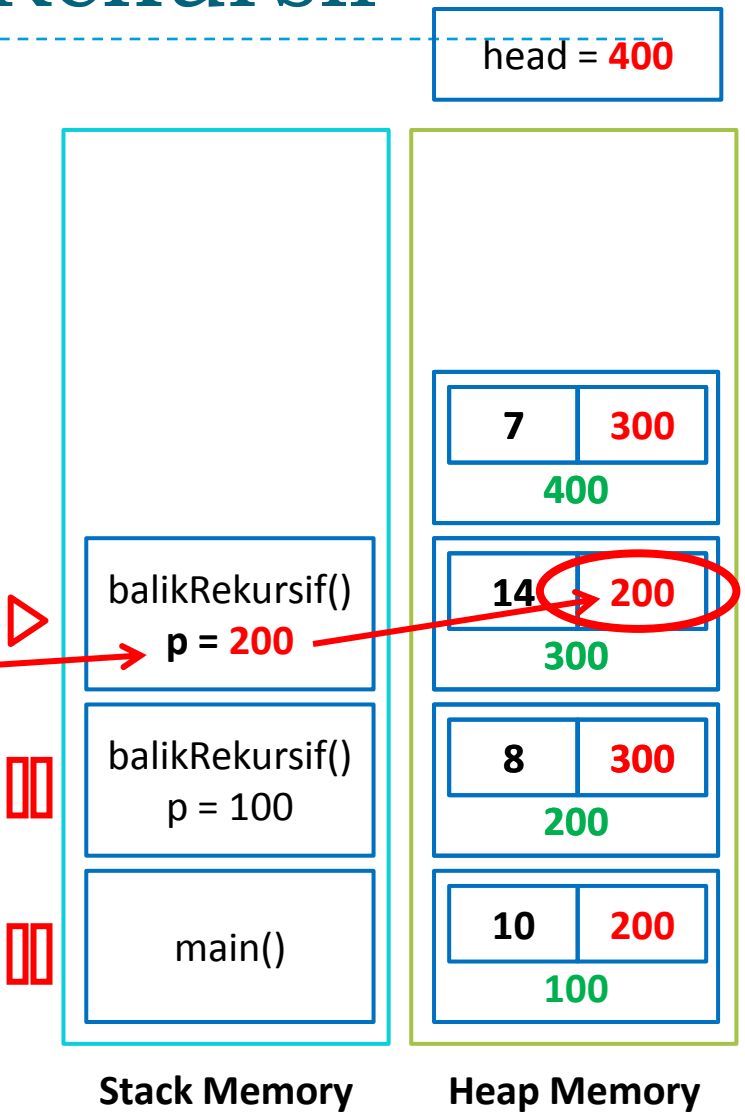


Membalik List secara Rekursif

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

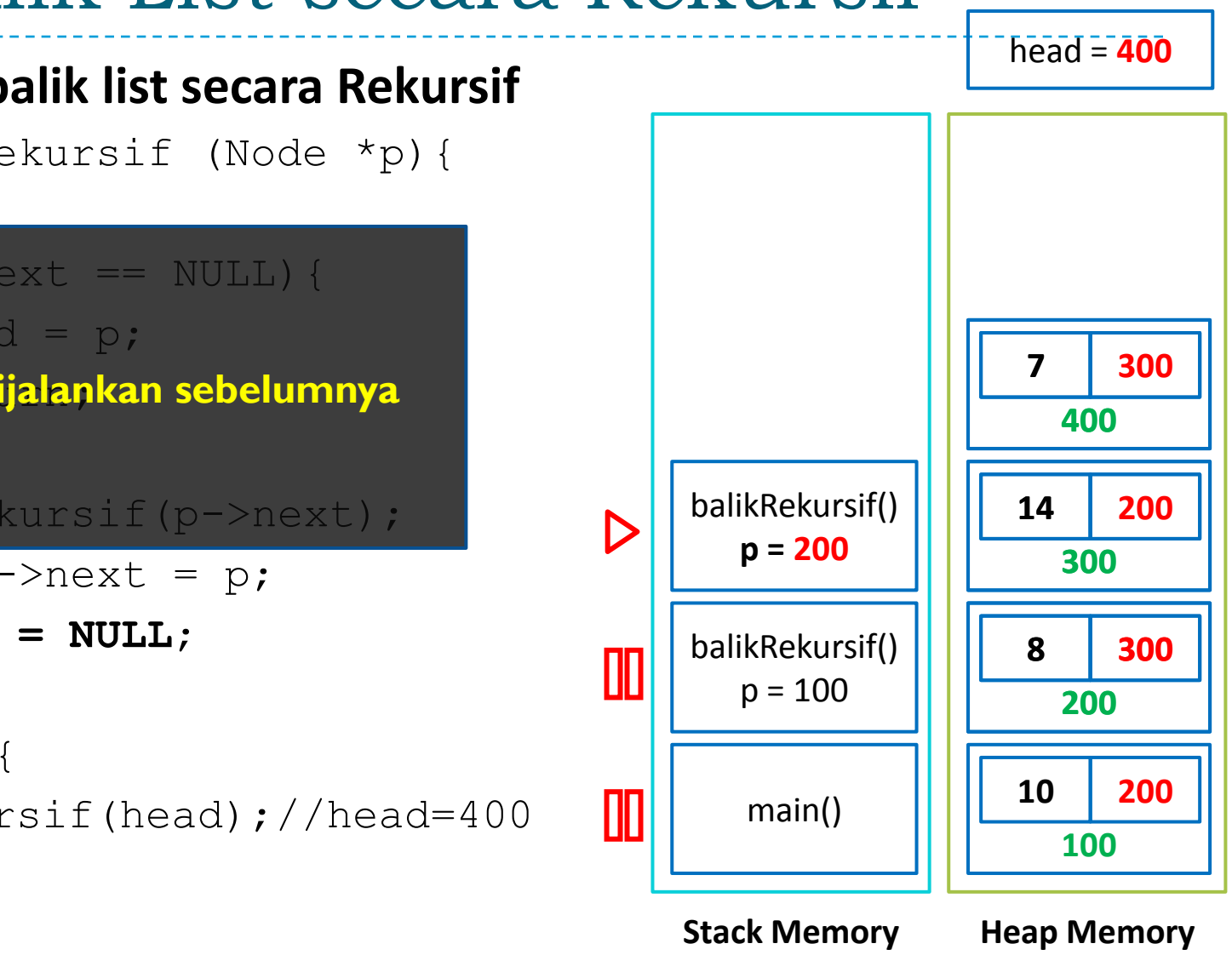


Membalik List secara Rekursif

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

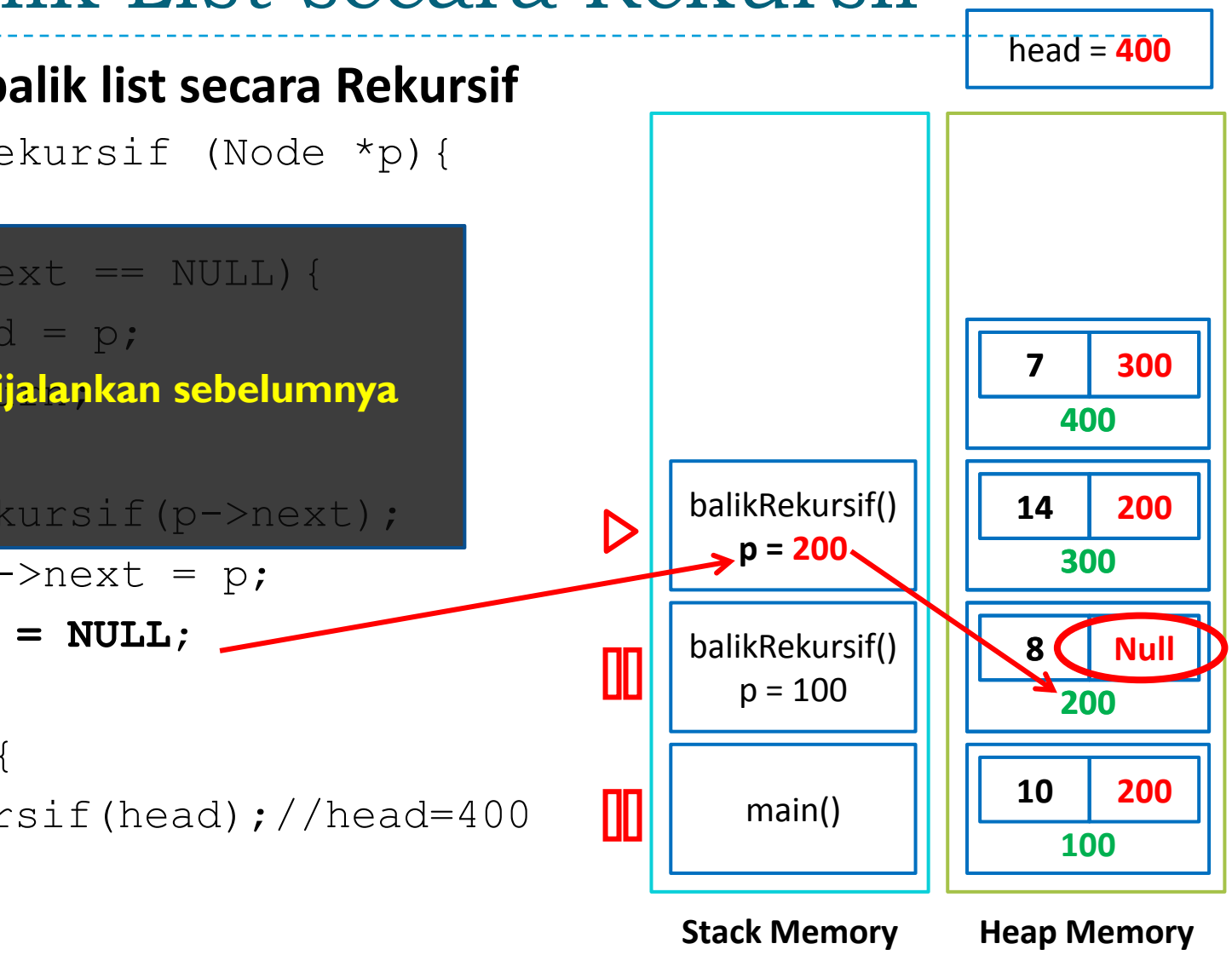


Membalik List secara Rekursif

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

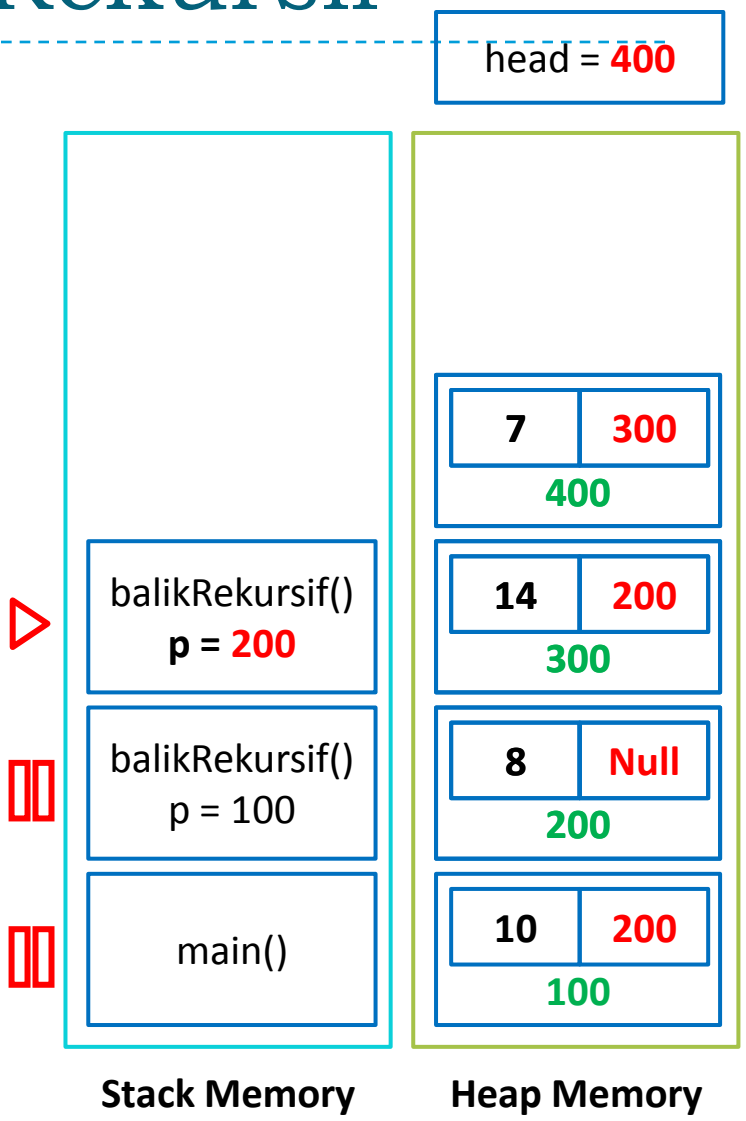


Membalik List secara Rekursif

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

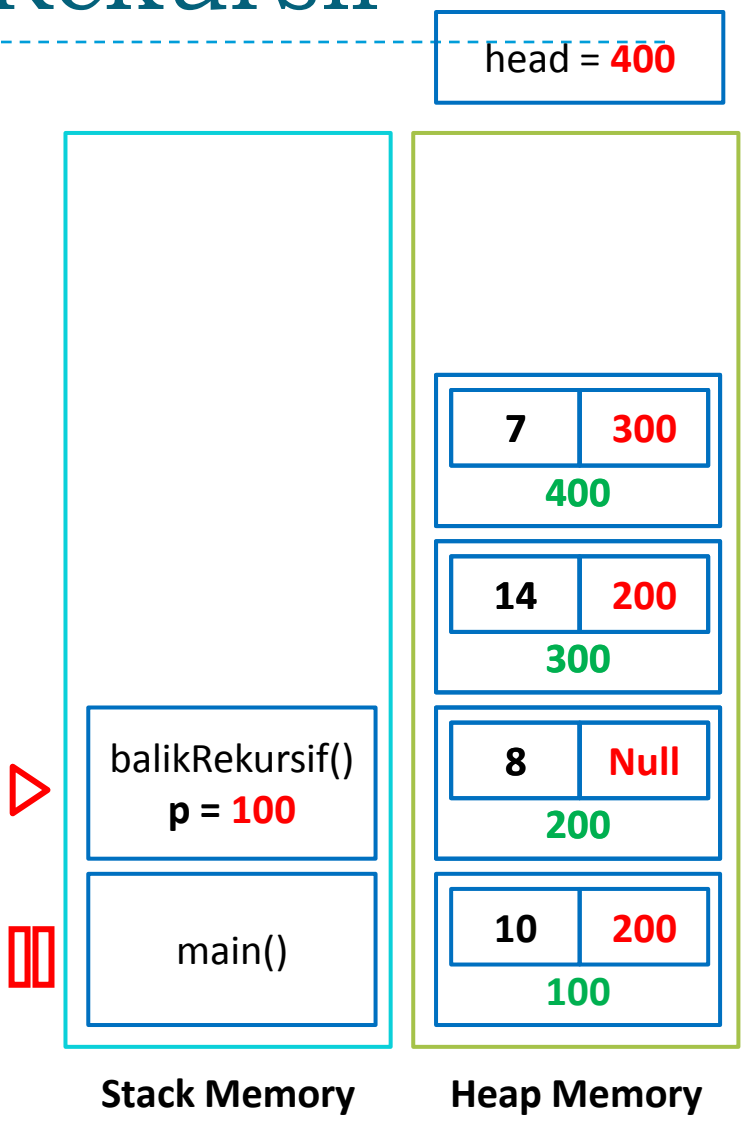


Membalik List secara Rekursif

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
}
```

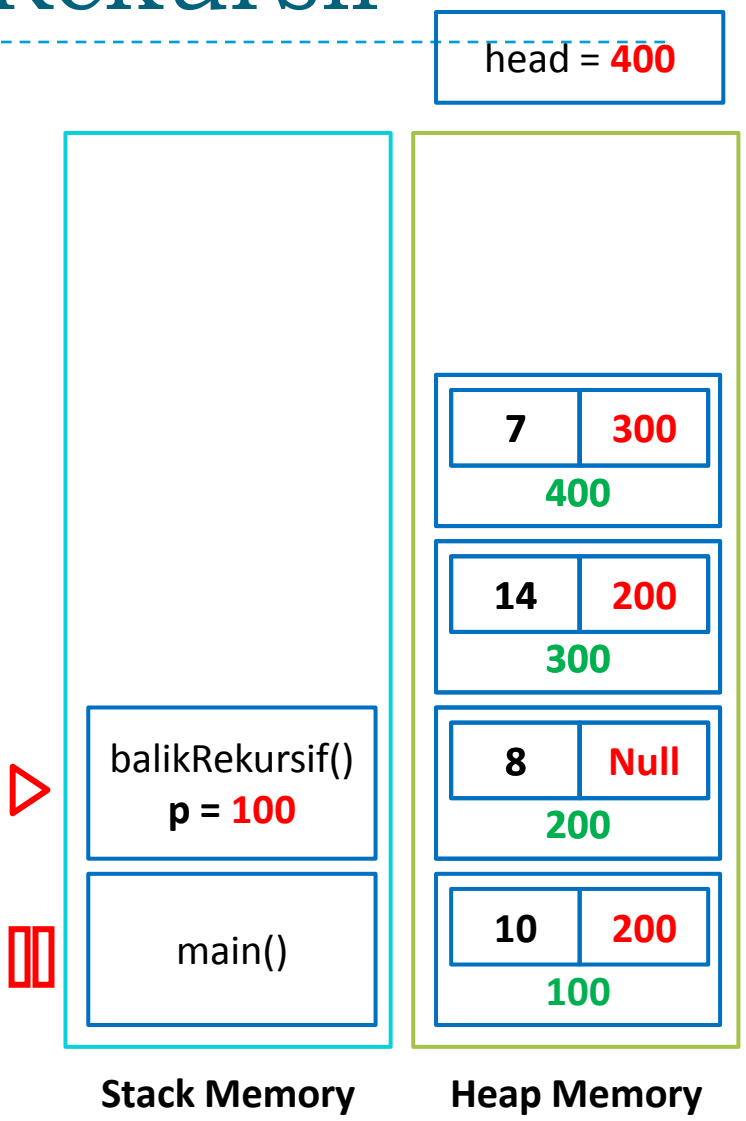


Membalik List secara Rekursif

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
}
```

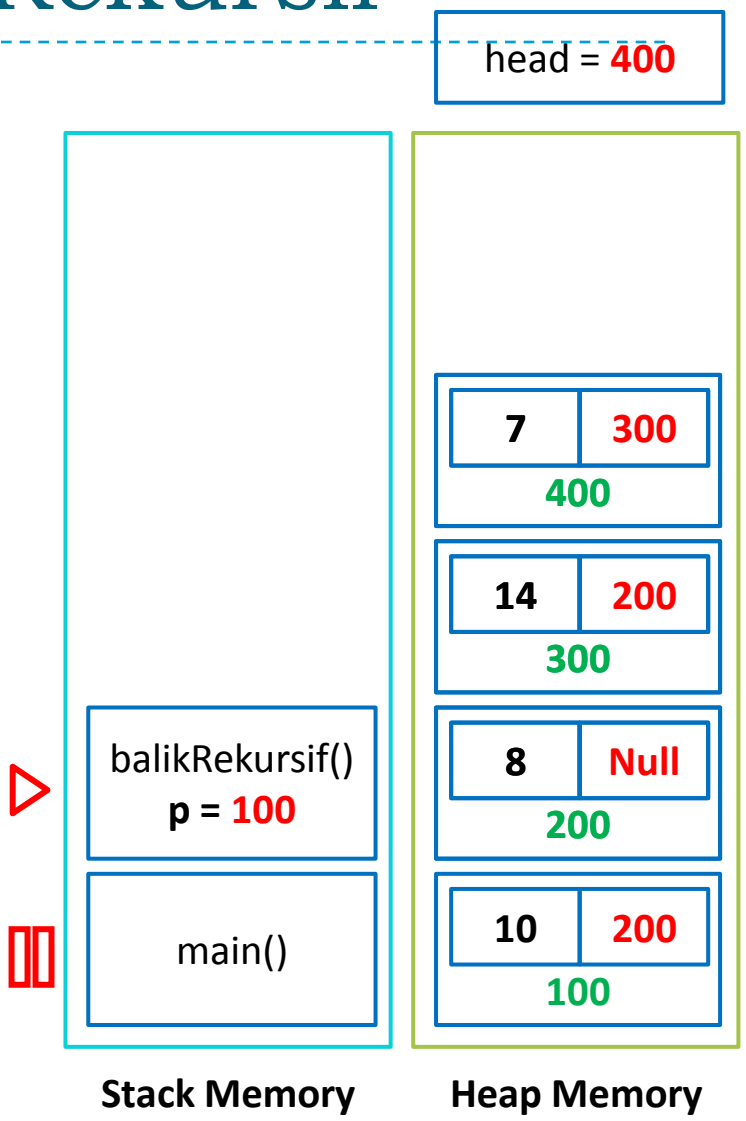


Membalik List secara Rekursif

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

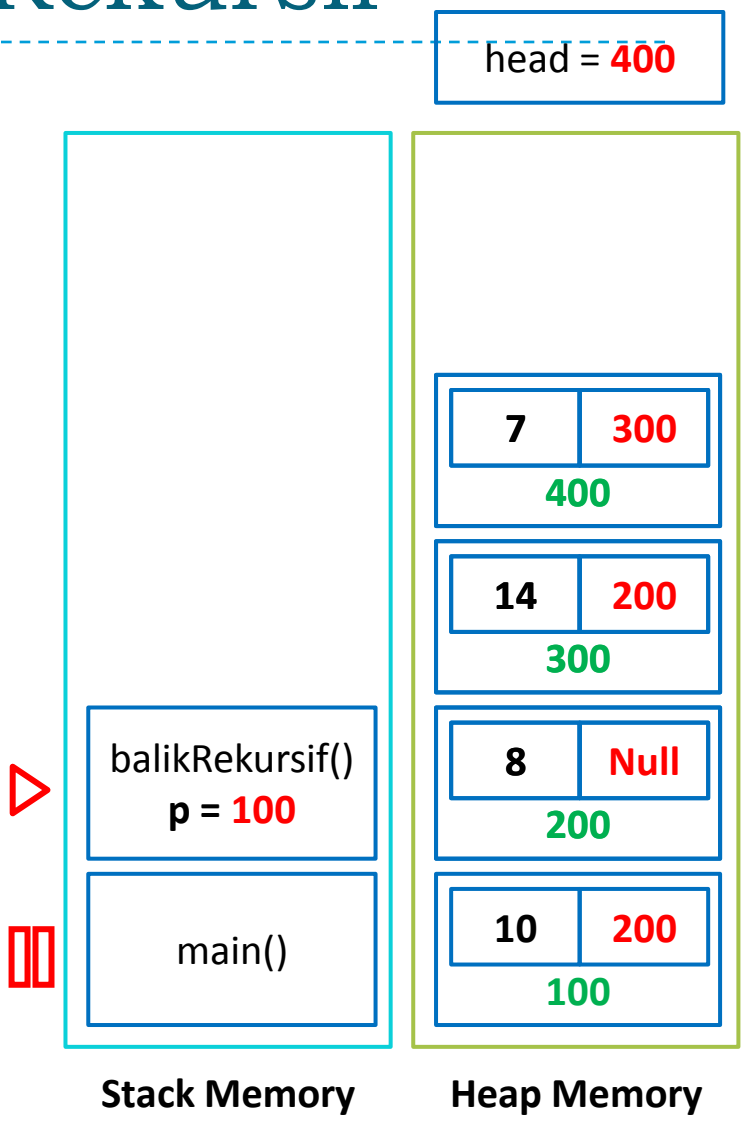


Membalik List secara Rekursif

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

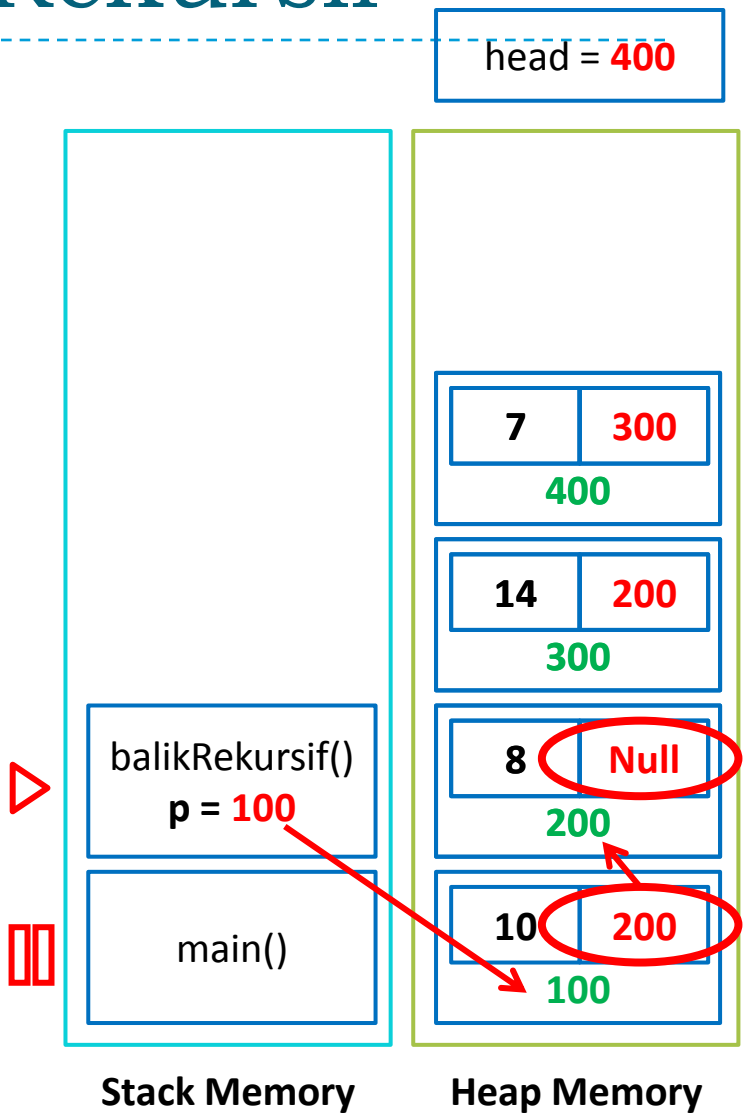


Membalik List secara Rekursif

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

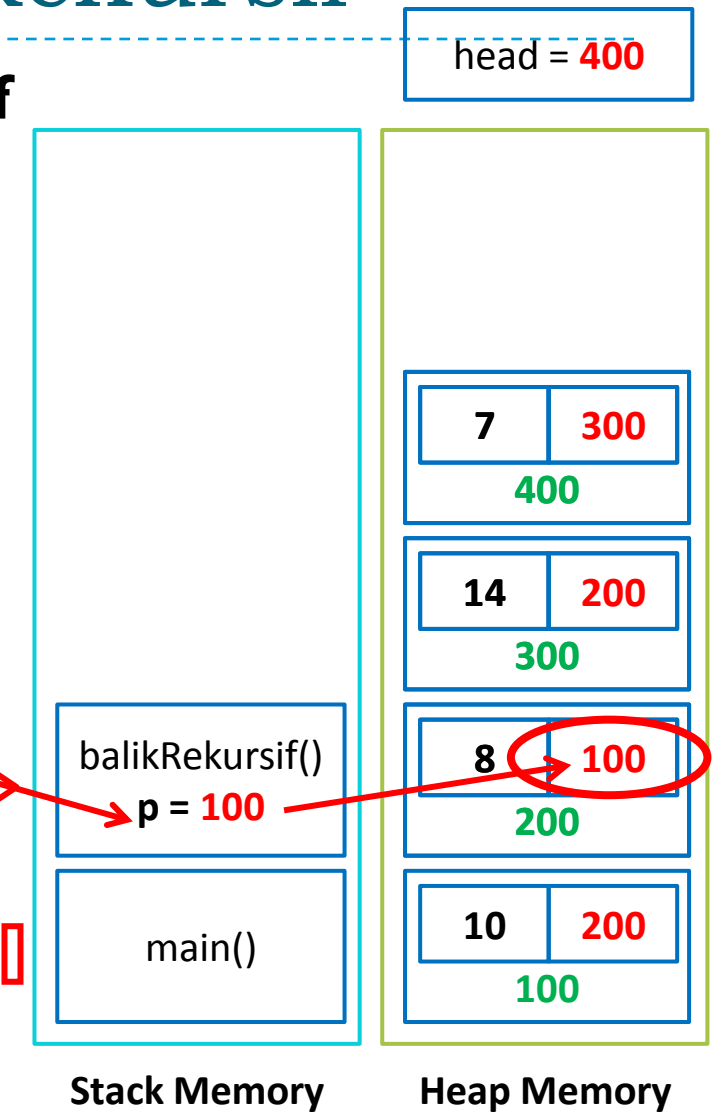


Membalik List secara Rekursif

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

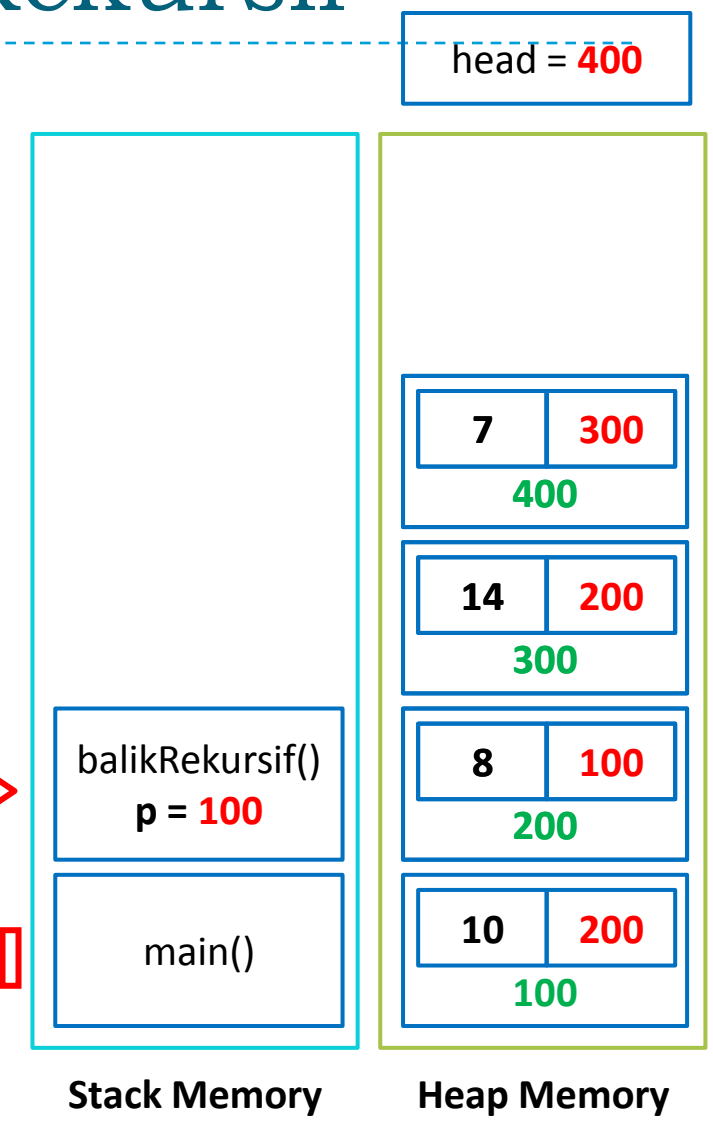


Membalik List secara Rekursif

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

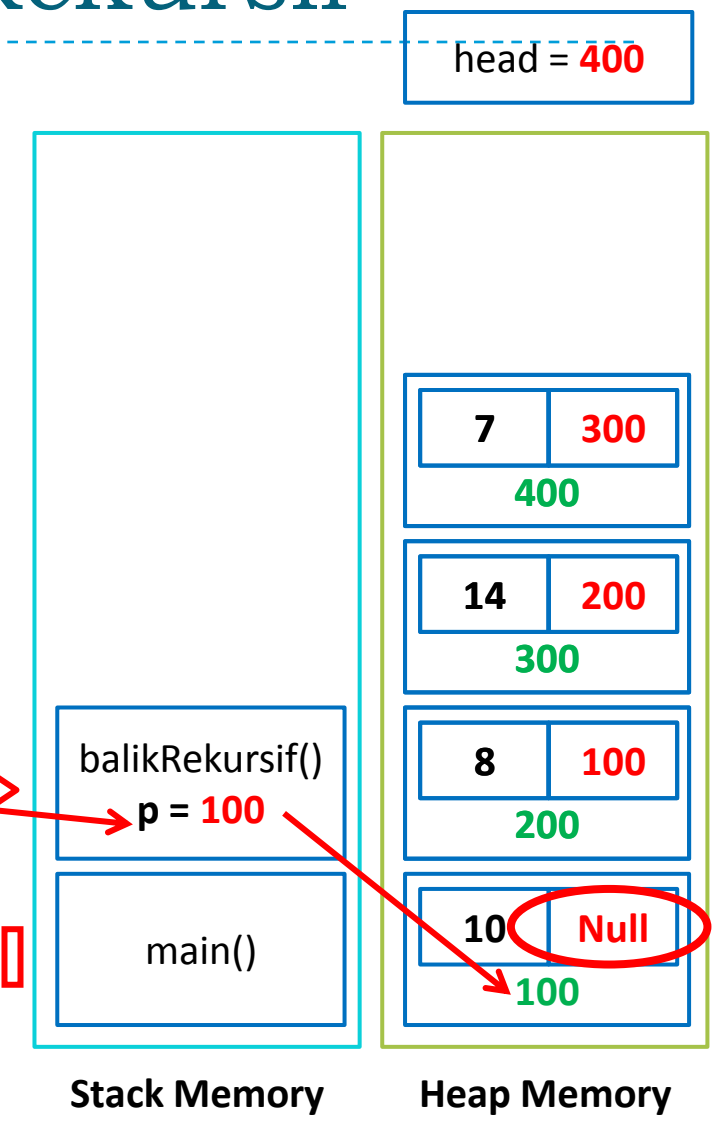


Membalik List secara Rekursif

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

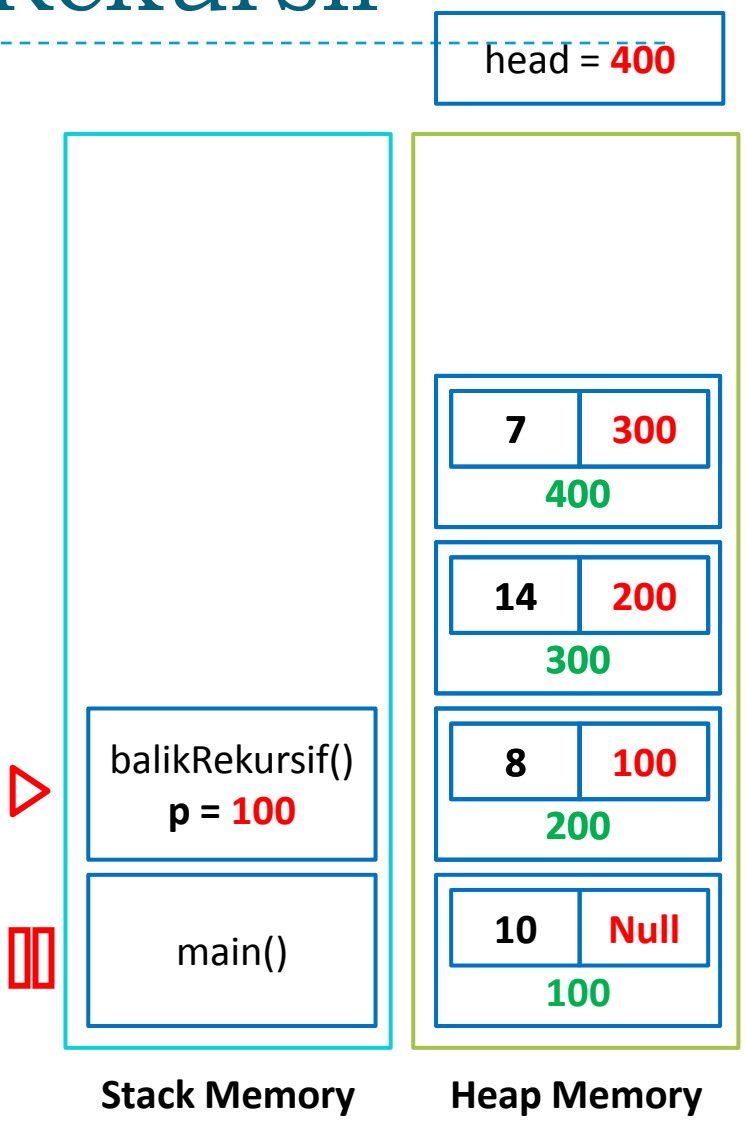


Membalik List secara Rekursif

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

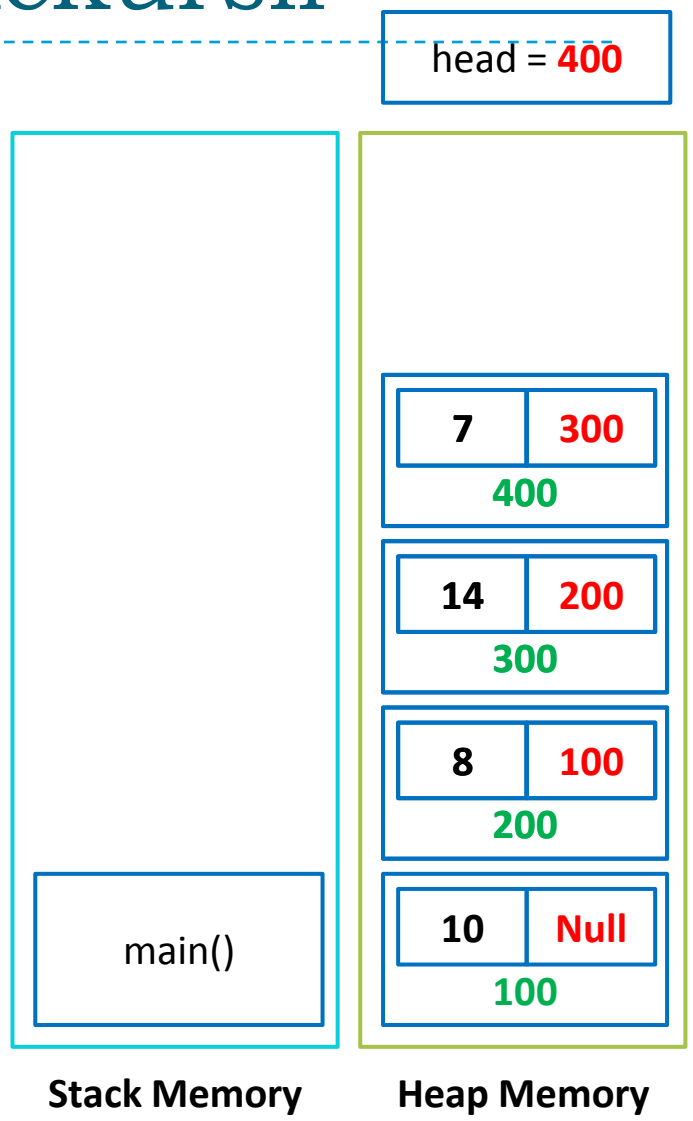


Membalik List secara Rekursif

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
}
```



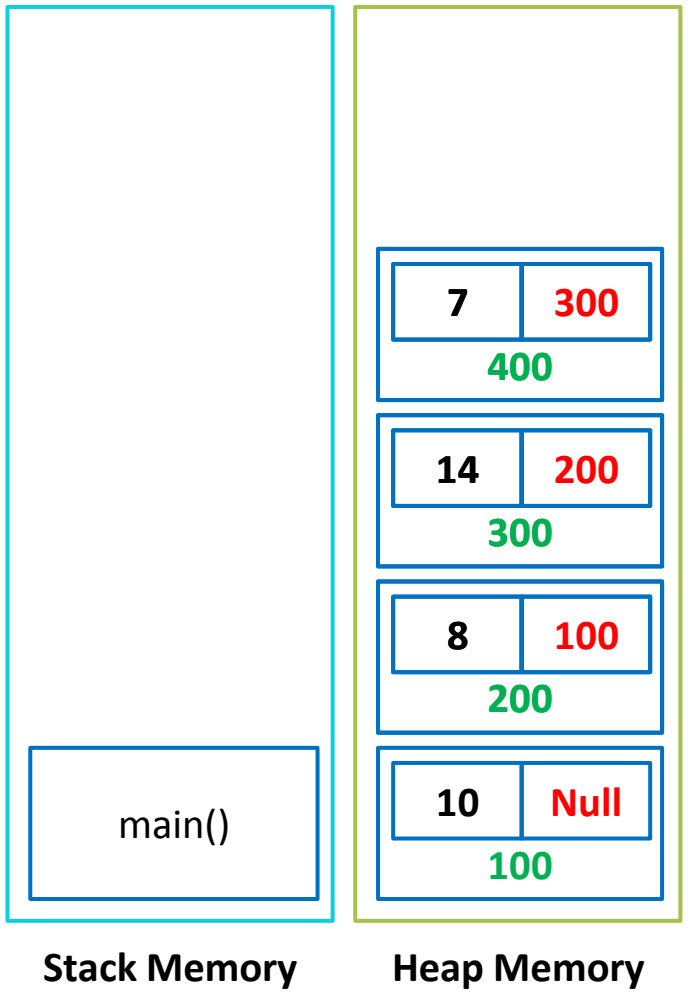
Membalik List secara Rekursif

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



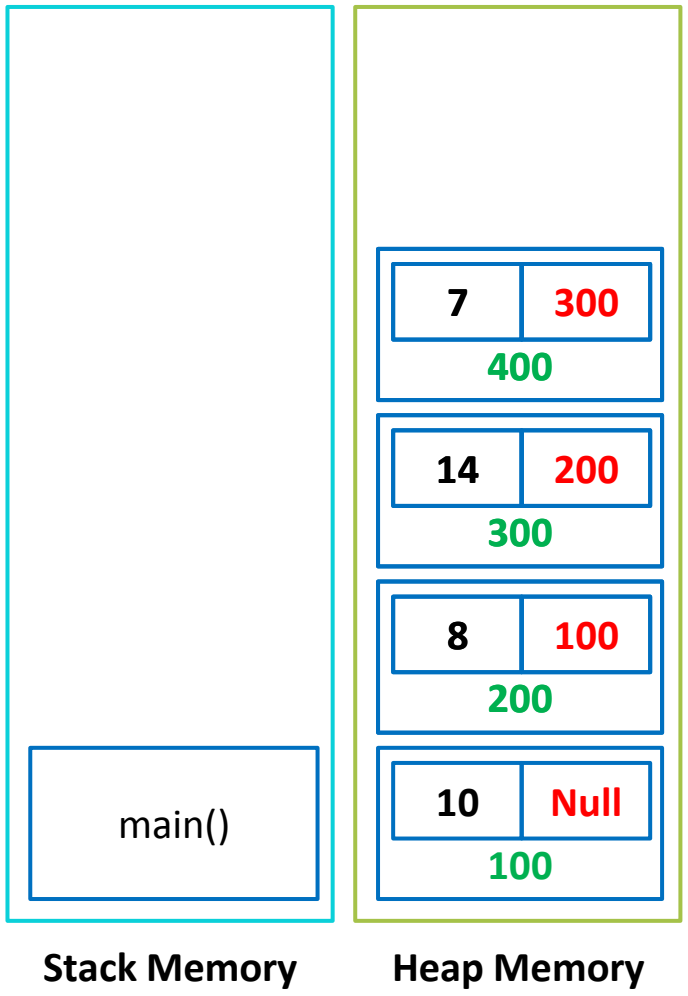
Membalik List secara Rekursif

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

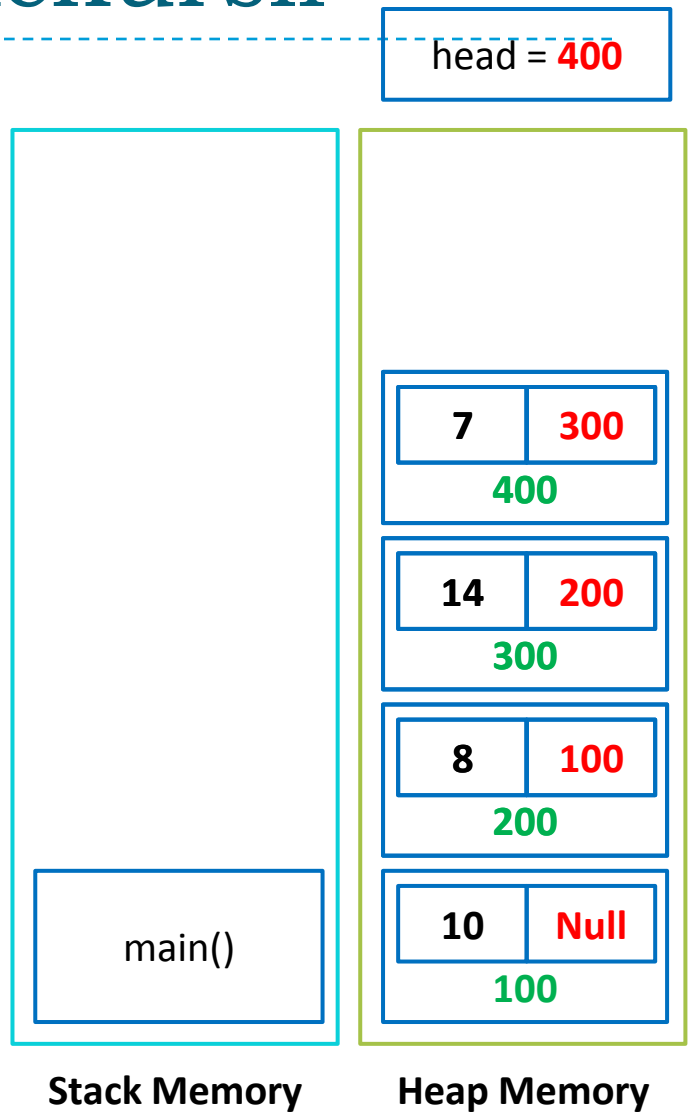


Membalik List secara Rekursif

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
}
```



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
}
```