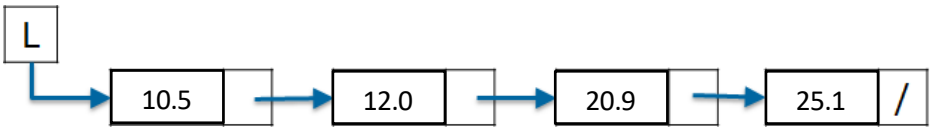
NAMA: Farrel Izaz Yuwono	NIM: 2311104014	KELAS: SE-07-01

1. Terdapat sebuah single linked list yang sudah terurut nilainya seperti di bawah ini:



a. Buatlah deklarasi list dari linked list di atas.

```
Struct node{
        Int info;
        Node* address
}
```

b. Buatlah prosedur untuk membuat sebuah list kosong.

```
Void listkosong(node*& head){
        head = nullptr;
}
```

c. Buatlah prosedur untuk membuat elemen baru bernama “elm” yang memiliki nilai “X” dimana “X” merupakan bilangan real.

```
Struct elm{
        Int X;
        Elm* address;
}

Int main(){
elm* elm1 = new elm();
   elm1->info = 10;
   elm1->address = nullptr;
}
```

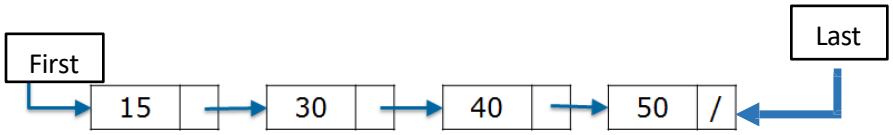
d. Buatlah prosedur insertAscending yang berisi algoritma untuk menambahkan data baru ke dalam list dengan aturan: data di dalam list harus selalu terurut secara menaik (ascending).

```
Prosedur insertAscending(X: float)
   1. Buat elemen baru newNode
     a. newNode.info ← X
     b. newNode.next ← null

   2. Jika (head = null) atau (head.info ≥ X)
maka:
     a. newNode.next ← head
     b. head ← newNode
```

```
    3. Jika tidak:
    a. newNode.next ← head.next
    b. head.next ← newNode
Akhir Prosedur
```

2. Terdapat sebuah single linked list seperti berikut:



a. Buatlah sebuah fungsi untuk mencari nilai X dalam list di atas.

```
search(int X) {
Node* temp = head;
while (temp != nullptr) {
        if (temp->info == X) {
        return true;
        }
        temp = temp->next;
}
return false;
}
```

b. Buatlah prosedur untuk menghapus elemen dengan kondisi berikut:
   1. Cek apakah list memiliki elemen bernilai X. Jika ada, maka:
      - Jika elemen yang berisi nilai X berada di awal atau di akhir list, maka hapus elemen tersebut.
      - Jika elemen X tidak berada di awal ataupun di akhir list, maka elemen yang dihapus adalah elemen setelah elemen bernilai X (contoh: X=30, maka elemen yang bernilai 40 yang dihapus).
   2. Jika tidak ada elemen yang bernilai X di dalam list, maka keluarkan notifikasi bahwa tidak ada elemen bernilai X dalam list.

```
void deleteElement(LinkedList& list, int X) {
   if (list.head == nullptr) {
      cout << "List kosong, tidak ada elemen untuk dihapus." << endl;
      return;
   }

   Node* temp = list.head;
   Node* prev = nullptr;


   if (temp != nullptr && temp->info == X) {

      if (temp->next == nullptr) {
         cout << "Menghapus elemen di awal: " << temp->info << endl;
         list.head = temp->next;  // Head diperbarui
         delete temp;
      } else {

         cout << "Menghapus elemen setelah " << temp->info << ": " << temp-
>next->info << endl;
         Node* toDelete = temp->next;
         temp->next = toDelete->next;
         delete toDelete;
      }
      return;
   }


   while (temp != nullptr && temp->info != X) {
      prev = temp;
      temp = temp->next;
   }


   if (temp == nullptr) {
      cout << "Elemen dengan nilai " << X << " tidak ditemukan dalam list." <<
endl;
      return;
   }


   if (temp->next == nullptr) {
      cout << "Menghapus elemen di akhir: " << temp->info << endl;
      prev->next = nullptr;
      delete temp;
   } else {
```

```
        cout << "Menghapus elemen setelah " << temp->info << ": " << temp->next->info << endl;
        Node* toDelete = temp->next;
        temp->next = toDelete->next;
        delete toDelete;
    }
}
```