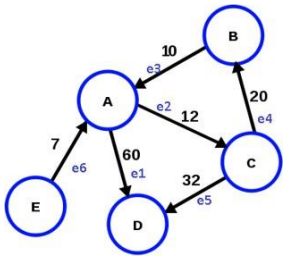
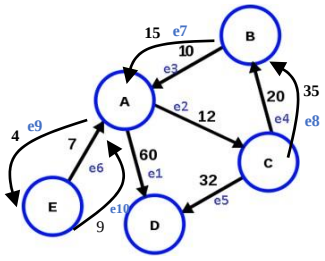
Soal Assessment CLO 3 – Kamis

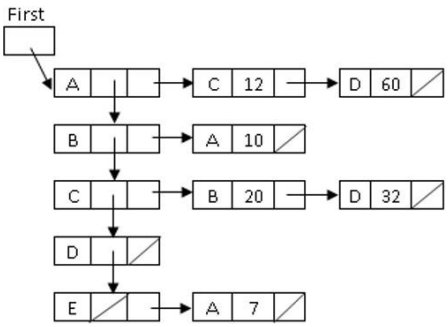
A. Graph (60 poin)

Terdapat sebuah graf ber-bobot ber-arah yang menggambarkan peta lokasi beserta jarak menuju setiap lokasi. Node melambangkan lokasi dan edge merupakan jalan menuju node lain. Di setiap edge terdapat jarak yang merupakan jarak yang harus ditempuh untuk menuju lokasi lain. Jalan menuju sebuah node bisa terdiri lebih dari 1 jalur (lebih dari 1 edge). Misal pada gambar graf kanan bawah, jarak dari kota B ke A terdapat 2 alternatif jalur, jalur e3 10 km jalur e7 15 km.





Adapun representasi graf yang digunakan adalah sbb (contoh gambar graf di kiri atas):



A. Buatlah pada isi dari graf.h dan graf.c berupa deklarasi struktur data dan primitif serta implementasi fungsi procedure graf di bawah ini: (60 poin)

Deklarasi yang digunakan:

(3 poin)

type adrNode : pointer to Node

type adrEdge : pointer to Edge

```
type Node <    idNode: char,
               nextNode: adrNode,
               firstEdge: adrEdge >

type Edge < id : char
            jarak : integer
            nextEdge: adrEdge >

type Graph < First : adrNode >
```

1. <u>procedure</u> createNode(**<u>input</u>** X char, **<u>output</u>** P : adrNode) **(2 poin)**

{IS. terdefinisi sebuah character X
FS. teralokasi sebuah elemen Node yang ditunjuk oleh P dengan info adalah X}

```
Kamus
Algoritma
        alokasi(P)
        idNode(P) <- X
        nextNode <- NIL
        firstEdge <- NIL
```

2. <u>procedure</u> createGraph(**<u>output</u>** G:Graph) **(2 poin)**

{IS. –
FS. terbentuk sebuah Graph G kosong dengan column dan row bernilai NIL}

```
Kamus
Algoritma
        First(G) <- NIL
```

3. Procedure showPeta (G : graph) (3 poin)

{IS. terdefinisi sebuah Graf yang mungkin kosong
FS. Tampil di layar detail data jalur antar node}
Ilustrasi :
A ke B : 4
A ke B : 6
A ke C : - B
ke A : 4 B
ke A : 10
B ke A : 6
B ke C : 4
dst...

```
Kamus
        N : adrNode
        E : adrEdge
Algoritma
        N ← First(G)
        if (N != NULL) then
                while (N != NULL) do
                    E=firstEdge(N)
                    while (E != NULL) do
                            print(idNode(N), " ke ", id(E), " : ", jarak(E))
                            E=nextEdge(E)
                    N=nextNode(N)

        else
                 print("GRAF KOSONG")
```

4. **Procedure InsertNewNode (Input/Output** G : Graph, **Input** idNode : char**) (10 poin)**

   {IS. terdefinisi sebuah Graph yang mungkin kosong dan sebuah id node yang akan di insertkan ke dalam Graph
   FS. Elemen node dengan info id Node menjadi elemen node terakhir pada list Node Graph G
   .....

5. **Function FindNode(**G: Graph, id_Node : char**)→** adrNode **(10 poin)**

   {terdapat Graph G yang mungkin kosong, kemudian akan dicari alamat dari node berinfo id_node, jika ditemukan maka akan dikembalikan alamat dari node tsb, jika tidak return NULL}
   .....

6. **Procedure Connecting (Input/Output** G : Graph, **Input** Node1, Node2: char, jarak: integer**) (10 poin)**

   {IS. terdefinisi sebuah Graph yang mungkin kosong, id Node 1 yang akan menunjuk ke arah Node 2 dan jarak dari node 1 ke node 2
   FS. Node 1 terhubung ke Node 2 oleh sebuah edge. Asumsi Node 1 dan node 2 pasti ada di Graph}
   .....

7. **Function biayaPerbaikanJalan** (G : Graph): integer **(10 poin)**

   {IS. terdefinisi sebuah Graph berarah berbobot yang mungkin kosong. Biaya perbaikan jalan per meter adalah 1 juta rupiah.
   FS. Dihasilkan total biaya perbaikan jalan untuk seluruh jalan yang ada di peta tsb. HINT : jumlahkan seluruh jarak pada seluruh edge}
   .....

B. Lengkapilah file main.cpp seperti perintah pada potongan program berikut (10 poin)

```
int main(){
// inisialisasi Graph G
...
cout<<"MEMBUAT NODE PADA GRAF"<<endl;
// TAMBAHKAN NODE A, B, C, D KE DALAM GRAPH
...
// TAMPILKAN SELURUH DATA NODE PADA GRAPH
...
cout<<"\nMEMBUAT EDGE PADA GRAF"<<endl;
// hubungkan A ke C dengan jarak 5 km , A ke D dengan jarak 10 km, B ke D dengan jarak 7km, dan D ke C dengan jarak 2 km
...

// hitung biaya perbaikan jalan dari total seluruh jalur yang ada
return 0;
```

B. Tree (40 poin)

1. ADT Binary Search Tree (BST) menggunakan Linked List sebagai berikut :

10 Poin

```
Type infotype: integer
Type address : pointer to node
Type node:     < info : infotype
                 left, right : address >
Type root: address
```

2. Fungsi/Procedur yang digunakan sebagai berikut :

20 Poin

```
function alokasi(x : infotype) → address
{IS. terdefinisi sebuah integer x
FS. Mengembalikan elemen Node baru dengan info = x, left dan right elemen = Nil}
Kamus
        P: address
Algoritma
        alokasi(P)
        info(P) ← x
        left(P) ← NIL
        right(P) ← NIL
        → P

procedure insertNode(in/out root : address, in x : infotype)
{IS. root mungkin kosong
FS. Mengalokasi Node baru dengan info=x dan dimasukan kedalam BST secara
rekursif dengan syarat jika x< info(root), maka Node baru menjadi anak kiri, jika
x>info(root), maka Node baru menjadi anak kanan}

procedure printInorder(in root : address)
{IS. root mungkin kosong
FS. Secara rekursif menampilkan isi dari Tree secara Inorder}
```

3. Tampilkan proses insert node dan hasil pre-ordernya dengan melengkapi file main.cpp !

```
Insert Node:
Berhasil insert node 9
Berhasil insert node 4
Berhasil insert node 8
Duplikat node 4
Berhasil insert node 7
Berhasil insert node 3
Berhasil insert node 10
Berhasil insert node 25
Print In Order: 3 4 7 8 9 10 25
```

10 Poin