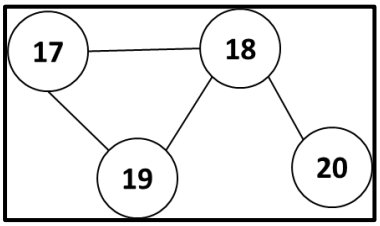
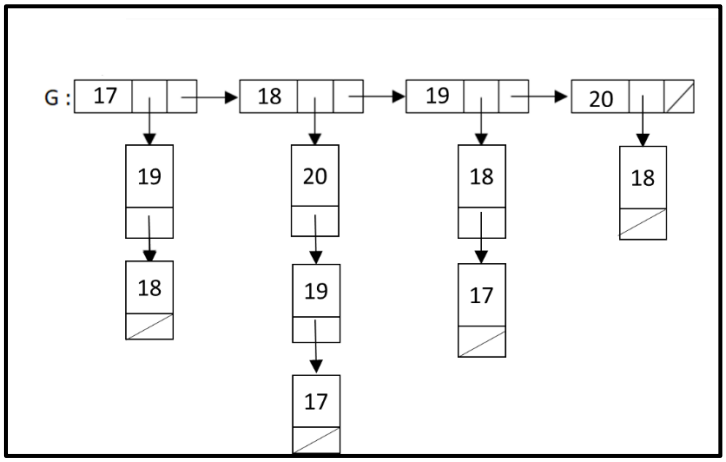
PRAKTIKUM MODUL 9
GRAPH

Diketahui sebuah Graph yang mana setiap nodenya menyimpan data berupa bilangan bulat. Graph tersebut direpresentasikan dalam list ketetanggan menggunakan sebuah Multi Linked List. Berikut ilustrasinya:





Buatlah ADT nya (Graph.h, Graph.cpp, dan Test_Graph.cpp)!

A. Spesifikasi (Silakan ditulis ulang dalam Bahasa C++)

```
type infotypeNode: integer
type addressNode: pointer to node
type addressNeighbour: pointer to neighbour
type node: <   info: infotypeNode
               next: addressNode
               firstNeighbour: addressNeighbour    >
type neighbour: <      info: infotypeNode
                       next: addressNeighbour      >
type Graph: addressNode

procedure createGraph (In/Out G: Graph)
function createNode (newInfo: infotypeNode) → addressNode
function createNeighbour (newInfo: infotypeNode) → addressNeighbour
procedure insertFirstNeighbour (In p: addressNeighbour, In pNode: addressNode)
procedure deleteFirstNeighbour (In pNode: addressNode, Out p: addressNeighbour)
procedure deleteAfterNeighbour (In prec: addressNeighbour, Out p: addressNeighbour)
procedure insertLastNode (In newInfo: infotypeNode, In/ Out G: Graph)
function searchNode (infoCari: infotypeNode, G: Graph) → addressNode
```

procedure connect (In node1, node2: infotypeNode, In G: Graph)
procedure searchNeighbour (In infoCari: infotypeNode, pN: addressNode) → addressNeighbour
procedure disconnect (In node1, node2: addressNode, In G: Graph)
procedure printGraph (In G: Graph)

B. Implementasi (Silakan ditulis ulang dalam Bahasa C++)

procedure createGraph (In/Out G: Graph)
{IS. –
FS. Terbentuk sebuah graph di mana G bernilai NIL.}
Kamus
Algoritma
G ← NULL

function createNode (newInfo: infotypeNode) → addressNode
{Return alamat alokasi memori sebuah node yang berisi newInfo.}
Kamus
p: addressNode
Algoritma
alokasi(p)
info(p) ← newInfo
next(p) ← NULL
firstNeighbour(p) ← NULL

function createNeighbour (newInfo: infotypeNode) → addressNeighbour
{Return alamat alokasi memori sebuah neighbour yang berisi newInfo.}
Kamus
p: addressNeighbour
Algoritma
alokasi(p)
info(p) ← newInfo
next(p) ← NULL

procedure insertFirstNeighbour (In p: addressNeighbour, In pNode: addressNode)
{IS. Terdefinisi pointer p berisi alamat neighbour dan pNode berisi alamat node. p≠NULL. pNode≠NULL. List tetangga node pNode mungkin kosong.
FS. Neighbour yang ditunjuk oleh p ditambahkan sebagai tetangga pertama node pNode.}
Kamus
Algoritma
next(p) ← firstNeighbour(pNode)
firstNeighbour(pNode) ← p

procedure deleteFirstNeighbour (In pNode: addressNode, Out p: addressNeighbour)
{IS. Terdefinisi pNode berisi alamat node (list tetangga pNode tidak kosong dan mungkin berisi satu elemen).
FS. p berisi alamat neighbour yang pertama, neighbour yang ditunjuk oleh p dihapus dari list tetangga pNode}

Kamus

Algoritma
p ← firstNeighbour(pNode)
firstNeighbour(pNode) ← next(p)
next(p) ← NULL

procedure deleteAfterNeighbour (In prec: addressNeighbour, Out p: addressNeighbour)
{IS. Terdefinisi pointer prec berisi alamat neighbour, prec bukan neighbour terakhir. List tetangga berisi lebih dari satu elemen.
FS. p berisi alamat neighbour setelah prec, neighbour yang ditunjuk oleh p dihapus dari list tetangga}
Kamus

Algoritma
p←next(prec)
next(prec)←next(p)
next(p)←NULL

C. Program Utama (Silakan ditulis ulang dalam Bahasa C++)

Kamus
G: Graph
Algoritma
createGraph(G)
printGraph(G) *{Graph Kosong}*
insertLastNode(17, G)
insertLastNode(18, G)
insertLastNode(19, G)
insertLastNode(20, G)
printGraph(G) *{Node 17: null; Node 18: null; Node 19: null; Node 20: null}*
connect(17,18,G)
printGraph(G) *{Node 17: 18; Node 18: 17; Node 19: null; Node 20: null}*
connect(17,19,G)
connect(18,19,G)
connect(18,20,G)

printGraph(G)	*{Node 17: 19, 18; Node 18: 20, 19, 17; Node 19: 18, 17; Node 20: 18}*
disconnect(18,20,G)	
printGraph(G)	*{Node 17: 19, 18; Node 18: 19, 17; Node 19: 18, 17; Node 20: null}*
disconnect(18,19,G)	
printGraph(G)	*{Node 17: 19, 18; Node 18: 17; Node 19: 17; Node 20: null}*
disconnect(18,17,G)	
printGraph(G)	*{Node 17: 19; Node 18: null; Node 19: 17; Node 20: null}*
disconnect(19,17,G)	
printGraph(G)	*{Node 17: null; Node 18: null; Node 19: null; Node 20: null}*

D. TUGAS ANDA

Buat implementasi procedure dan function berikut ini:

procedure insertLastNode (In newInfo: infotypeNode, In/ Out G: Graph)
{IS: Graph mungkin kosong
FS: Node yang berisi newInfo disisipkan sebagai elemen terakhir jika dalam graph belum ada node yang infonya = newInfo.}

function searchNode (infoCari: infotypeNode, G: Graph) → addressNode
{Graph tidak kosong. Return alamat node jika ditemukan, return NULL jika tidak ditemukan.}

procedure connect (In node1, node2: infotypeNode, In G: Graph)
{Menghubungkan (membuat sisi) antara dua buah node pada graph tidak berarah.
IS: Graph mungkin kosong. Node 1 atau node 2 mungkin tidak ada.
FS: Neighbour yang berisi info = node 1 disisipkan sebagai neighbour pertama node 2 dan neighbour yang berisi info = node 2 disisipkan sebagai neighbour pertama node 1, jika kedua node ada dalam graph.}

procedure searchNeighbour (In infoCari: infotypeNode, pN: addressNode) → addressNeighbour
{List tetangga node pN tidak kosong. Return alamat neighbour jika ditemukan, return NULL jika tidak ditemukan.}

procedure disconnect (In node1, node2: addressNode, In G: Graph)
{Menghapus sisi antara dua buah node yang terhubung pada graph tidak berarah.
IS: Graph mungkin kosong. Node 1 atau node 2 mungkin tidak ada. Node 1 dan node 2 mungkin tidak terhubung.
FS: Neighbour yang berisi info = node 1 dihapus dari list tetangga node 2 dan neighbour yang berisi info = node 2 dihapus dari list tetangga node 1, jika kedua node ada dalam graph dan terhubung.}

procedure printGraph (In G: Graph)
{IS: Graph mungkin kosong
FS: Menampilkan graph dalam bentuk info list ketetanggaan.}