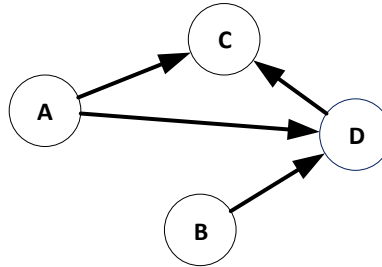


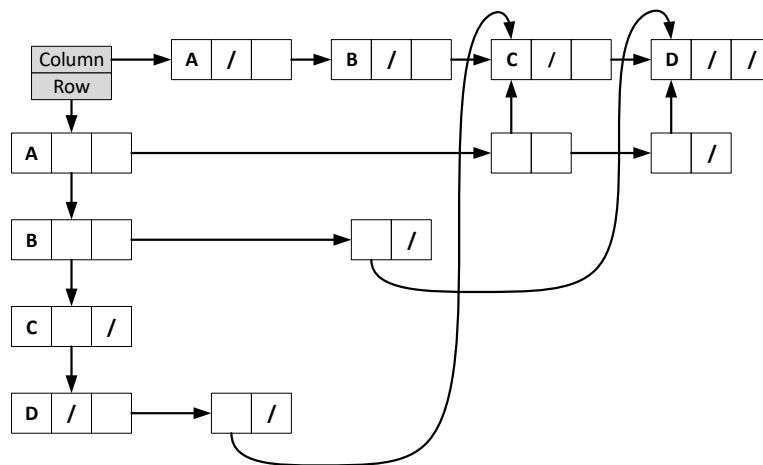
Soal Praktikum Modul 10 (Assessment CLO3 : Graph) – Versi 5

GRAPH - STRUKTUR DATA – CCH1D4

Sebuah Multi Linked-List digunakan untuk merepresentasikan sebuah Graf Berarah seperti contoh berikut ini!



Gambar 1. Contoh Graph Berarah



Gambar 2. Contoh Multi Linked-List untuk Graph Berarah pada gambar 1

Pada gambar 2 terlihat graf tersusun seperti matrik ketetanggaan, di mana terdapat dua list dari node/simpul, dengan nilai first pada masing-masing list tersimpan pada pointer row dan column. node A bertetangga dengan node C dan node D, sedangkan node B bertetangga dengan node D.

Buatlah sebuah program dengan menggunakan Code::Block yang mengimplentasikan Graf Berarah tersebut sesuai dengan petunjuk berikut ini!

1. Buatlah file GRAPH.h yang berisi spesifikasi dari ADT graph seperti gambar 1 dan 2!

```
type adrNode :  
type adrEdge :  
type Node < ... >  
type Edge < ... >  
type Graph < column,row : adrNode >
```

Soal Praktikum Modul 10 (Assessment CLO3 : Graph) – Versi 5

GRAPH - STRUKTUR DATA – CCH1D4

```
procedure createNode(input X char, output P : adrNode)
{IS. terdefinisi sebuah character X
 FS. teralokasi sebuah elemen Node yang ditunjuk oleh P dengan info adalah X}

procedure createGraph(output G:Graph)
{IS. -
 FS. terbentuk sebuah Graph G kosong dengan column dan row bernilai NIL}

procedure insertLast(input/output first:adrNode, input P : adrNode)
{IS. terdefinisi sebuah first dari list node (mungkin NIL), dan sebuah Node yang
 disimpan pada pointer P
 FS. data baru yang ditunjuk oleh P ditambahkan sebagai elemen terakhir dari list}

procedure addNode(input/output G:Graph, input X : char)
{IS. terdefinisi sebuah Graph G (mungkin kosong) dan sebuah character X
 FS. terbentuk node dengan info X, kemudian ditambahkan secara insert last pada
 list node dengan first row(G) dan column(G). Petunjuk: alokasi dan insert dilakukan
 dua kali}

function findNode(first: adrNode, X : char) → adrNode
{mengembalikan alamat memori dari elemen node yang memiliki info == X pada list
 node, atau NIL apabila X tidak ditemukan di dalam list}

procedure connect(input/output G:Graph, input X,Y : char)
{IS. terdefinisi sebuah Graph G (mungkin kosong) dan character X dan Y
 FS. menambahkan edge pada node berinfo X dan menghubungkan dengan node berinfo Y.
 Alokasi elemen edge dilakukan di sini, dan edge ditambahkan sebagai elemen pertama
 pada list edge}

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

function inDegree(G:Graph, V:char) → integer
{Mengembalikan In Degree atau Derajat Masuk pada node V dalam Graph G (Graph tidak
 kosong)}
```

2. Lengkapi implementasi dari spesifikasi yang telah dibuat sebelumnya pada sebuah file GRAPH.cpp!
3. Lengkapi file main.cpp seperti perintah pada potongan program berikut ini!

```
int main(){
    // inisialisasi Graph G
    ...
    cout<<"MEMBUAT NODE PADA GRAF"<<endl;
    // TAMBAHKAN NODE A, B, C, D KE DALAM GRAPH
    ...
    // TAMPILKAN ISI GRAPH
    ...
    cout<<"\\nMEMBUAT EDGE PADA GRAF"<<endl;
```

Soal Praktikum Modul 10 (Assessment CLO3 : Graph) – Versi 5

GRAPH - STRUKTUR DATA – CCH1D4

```
// hubungkan A ke C, A ke D, B ke D, dan D ke C
...
return 0;
}
```

```
MEMBUAT NODE PADA GRAF
A -
B -
C -
D -

MEMBUAT EDGE PADA GRAF
A - D - C -
B - D -
C -
D - C -
```

Gambar 3. Tampilan program setelah dijalankan

4. Tambahkan code berikut ini pada bagian akhir dari file main.cpp!

```
cout<<"\nMENAMBAHKAN NODE LAIN PADA GRAF"<<endl;
addNode(G, 'A');
addNode(G, 'B');
addNode(G, 'E');
addNode(G, 'C');
addNode(G, 'H');
addNode(G, 'D');
addNode(G, 'F');
printGraph(G);
cout<<"\nMENAMBAHKAN EDGE BARU PADA GRAF"<<endl;
connect(G, 'A', 'E');
connect(G, 'E', 'B');
connect(G, 'H', 'B');
connect(G, 'F', 'D');
connect(G, 'C', 'F');
connect(G, 'C', 'E');
connect(G, 'D', 'H');
connect(G, 'E', 'H');
connect(G, 'F', 'A');
printGraph(G);
```

```
MENAMBAHKAN NODE LAIN PADA GRAF
A - D - C -
B - D -
C -
D - C -
E -
H -
F -

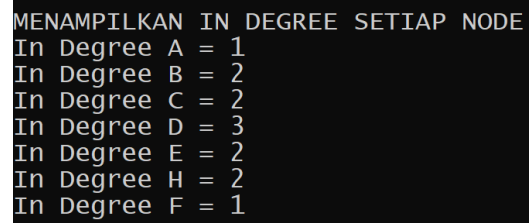
MENAMBAHKAN EDGE BARU PADA GRAF
A - E - D - C -
B - D -
C - E - F -
D - H - C -
E - H - B -
H - B -
F - A - D -
```

Gambar 4. Tampilan program setelah dijalankan dengan tambahan node dan edge

Soal Praktikum Modul 10 (Assessment CLO3 : Graph) – Versi 5

GRAPH - STRUKTUR DATA – CCH1D4

5. Tampilkan In Degree setiap node pada Graph pada file main.cpp!



```
MENAMPILKAN IN DEGREE SETIAP NODE
In Degree A = 1
In Degree B = 2
In Degree C = 2
In Degree D = 3
In Degree E = 2
In Degree H = 2
In Degree F = 1
```

Gambar 5. Tampilan program ketika menampilkan in degree setiap node