**SOAL PRAKTIKUM KOM120H STRUKTUR DATA**
**PERTEMUAN 8 STRUKTUR DATA *GRAPH***

Berikut ini adalah potongan program yang digunakan untuk membuat representasi *graph* **berarah** (*directed graph*) menggunakan struktur data *adjacency matrix*. Program tersebut mendefinisikan graph sebagai sebuah **struct** dengan elemen berupa banyaknya simpul (**n_vertices**), banyaknya sisi (**n_edges**), dan sebuah matrix sebagai *adjacency matrix* dari *graph* tersebut. Di sini diasumsikan banyaknya simpul dalam *graph* selalu ≤ 100.

Program ini kemudian membaca sebuah *file* input yang merepresentasikan *graph* yang ingin dibangun *adjacency matrix*-nya, dengan format sebagai berikut:

- baris pertama *file* input berisi dua buah bilangan **n_v** dan **n_e**, menyatakan banyaknya simpul dan banyaknya sisi pada *graph*. Kita asumsikan kemudian bahwa simpul-simpul *graph* akan diberi nama berupa angka dari **0, 1, …, (n_v-1).**
- Sebanyak **n_e** baris berikutnya akan berisi dua buah bilangan **a** dan **b** masing-masing bernilai antara **0** dan **n_v - 1**, yang menyatakan bahwa ada sisi antara simpul nomor **a** dan **b**
-

```c
#include <stdio.h>

#define MAXNUM_VERTICES 100

/*!
Struktur data untuk merepresentasikan sebuah graf dengan
menggunakan adjacency matrix.
Graph akan memiliki sebanyak n_vertices simpul dan n_edges sisi.
Kita mengasumsikan bahwa graph akan memiliki maksimal hanya MAXNUM_VERTICES simpul.
*/
typedef struct
{
    int n_vertices;
    int n_edges;
    int adjacency_matrix[MAXNUM_VERTICES][MAXNUM_VERTICES];
} Graph ;
```

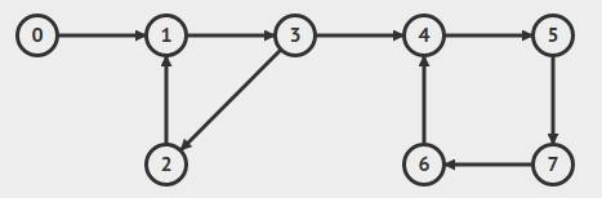
```c
/*
Fungsi untuk menginisialisasi adjacency matrix graph dg cara mengisi
sub-matrix kiri atas dg ukuran n_vertices x n_vertices dari adjacency matrix
dg nilai 0, sedangkan sisanya (yang tidak relevan untuk tujuan representasi graph
dg jumlah vertex sebanyak n_vertices) akan diisi dg nilai -1.
*/
void inisialisasi_graph(Graph *g, int n_v, int n_e)
{
    int i,j;

    g->n_vertices = n_v;
    g->n_edges = n_e;

    for(i = 0; i < MAXNUM_VERTICES; i++)
    {
        for(j = 0; j < MAXNUM_VERTICES; j++)
        {
            if(i < n_v && j < n_v)
            /* Soal no. 1: lengkapi untuk menyimpan nilai 0 pada sub-matrix kiri-atas dari adjacency matrix*/
            else
            /* Soal no. 2: lengkapi untuk menyimpan nilai -1 pada elemen-elemen sisanya */
        }
    }
}
```

```c
/*!
Fungsi untuk mencetak adjacency matrix dari sebuah graph g
*/
void print_adjacency_matrix(Graph *g)
{
    int i,j;
    for(i = 0; i < g->n_vertices; i++)
        printf("\t%d", i);
    printf("\n");

    for(i = 0; i < g->n_vertices; i++)
    {
        printf("%d", i);
        for(j = 0; j < g->n_vertices; j++)
        {
            printf("\t%d", g->adjacency_matrix[i][j]);
        }
        printf("\n");
    }
}
```

```c
int main()
{
    int n_v = 0;
    int n_e = 0;
    int i,j;
    scanf("%d %d", &n_v, &n_e);

    Graph g;
    inisialisasi_graph(&g, n_v, n_e);

    for(i = 0; i < n_e; i++)
    {
        int a, b;
        scanf("%d %d", &a, &b);
        /* Soal 3 : lengkapi agar adjacency matrix pada g menyimpan nilai 1 pada baris a kolom b */
    }

    print_adjacency_matrix(&g);

    return 0;
}
```

Program kemudian pertama-tama akan menginisialisasi *adjacency matrix* dengan menggunakan nilai 0 untuk semua elemen *sub-matrix* kiri-atas berukuran **n_v x n_v**. Sisanya akan diisi dengan nilai -1 untuk menunjukkan bahwa mereka tidak relevan untuk representasi *graph*. Kemudian, program akan membaca file input graph dan membuat represntasi *adjacency matrix*-nya. Terakhir, program akan menampilkan *adjacency matrix*.

Sebagai contoh, untuk *graph* berarah sebagai berikut:




*file* input yang dibutuhkan adalah sebagai berikut:

```
8 9
0 1
1 3
3 2
2 1
3 4
4 5
5 7
7 6
6 4
```

Jika kita *compile* program yang telah dilengkapi menjadi sebuah file ***executable*** yang bernama ***adjacency*,** dan *file* input di atas bernama **input.txt**, kita dapat menjalankan program tersebut sebagai berikut:

**Windows:**
C:\> adjacency < input.txt
**Linux:**
$ ./adjacency < input.txt maka kita seharusnya mendapatkan

*Output* program seperti berikut:

```
   0  1  2  3  4  5  6  7
0  0  1  0  0  0  0  0  0
1  0  0  0  1  0  0  0  0
2  0  1  0  0  0  0  0  0
3  0  0  1  0  1  0  0  0
4  0  0  0  0  0  1  0  0
5  0  0  0  0  0  0  0  1
6  0  0  0  0  1  0  0  0
7  0  0  0  0  0  0  1  0
```

**Soal**

1. Tuliskan baris yang harus diisikan pada bagian yang ditandai soal 1 pada potongan program di atas!
2. Tuliskan baris yang harus diisikan pada bagian yang ditandai soal 2 pada potongan program di atas!
3. Tuliskan baris yang harus diisikan pada bagian yang ditandai soal 3 pada potongan program di atas!
4. Bagaimana jika kita hanya ingin merepresentasikan *graph* tidak berarah? Ingat bahwa pada *graph* tidak berarah, *adjacency matrix*-nya adalah sebuah matriks simetrik. Tuliskan modifikasi pada baris pada soal no. 3 jika *graph*-nya tidak berarah!

Potongan program di bawah ini mengimplementasikan algoritme DFS dengan menggunakan representasi *adjacency matrix* yang digunakan sebelumnya.

```
typedef enum {WHITE, GRAY, BLACK} COLOR;
/*!
Prosedur yang mengimplementasikan DFS_visit.
*/
void DFS_visit(Graph *g, COLOR *vertex_colors, int v)
{
    int i;
    // Tampilkan v
    printf("%d ", v);

    // Tandai v dengan warna GRAY
    vertex_colors[v] = GRAY;

    // Cari vertex yang adjacent terhadap v, jika masih WHITE, panggil DFS_visit vertex itu
    for (i = 0; i < g->n_vertices; i++)
    {
        /* Soal 5: Lengkapi agar DFS_visit dipanggil pada vertex i yang
                   adjacent terhadap v dan masih berwarna WHITE
        */
    }

    vertex_colors[v] = BLACK;
}
```

```
/*!
Prosedur yang mengimplementasikan DFS.
*/
void DFS(Graph *g)
{
    COLOR vertex_colors[MAXNUM_VERTICES];
    int i;
    for (i = 0; i < g->n_vertices; i++)
        vertex_colors[i] = WHITE;

    for (i = 0; i < g->n_vertices; i++)
    {
        if(vertex_colors[i] == WHITE)
            DFS_visit(g, vertex_colors, i);
    }

    printf("\n");
}
```

5. Tuliskan baris program yang harus dilengkapi pada bagian yang tertera di atas agar DFS_visit() dapat dipanggil pada semua *vertex* yang *adjacent* terhadap **v** dan masih berwarna WHITE.
6. Urutan *vertex* apakah yang akan muncul jika program yang telah dilengkapi di atas diterapkan pada contoh *graph* yang diberikan di atas?