Nomor 1-3 (Full Codes)

```c
#include <stdio.h>
#define MAXNUM_VERTICES 100

typedef struct {
    int n_vertices;
    int n_edges;
    int adjacency_matrix[MAXNUM_VERTICES][MAXNUM_VERTICES];
} Graph;

void init_graph(Graph *g, int n_vertices, int n_edges) {
    int i, j;

    g->n_vertices = n_vertices;
    g->n_edges = n_edges;

    for(i = 0; i < MAXNUM_VERTICES; i++) {

        for(j = 0; j < MAXNUM_VERTICES; j++) {
            if(i < n_vertices && j < n_vertices) {
                g->adjacency_matrix[i][j] = 0;
            } else {
                g->adjacency_matrix[i][j] = -1;
            }
        }
    }
}

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");
    }
}

int main() {
    int n_vertices = 0;
    int n_edges = 0;
    int i;
    scanf("%d %d", &n_vertices, &n_edges);

    Graph g;
    init_graph(&g, n_vertices, n_edges);

    for(i = 0; i < n_edges; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        g.adjacency_matrix[a][b] = 1;
        // g.adjacency_matrix[b][a] = 1;  // tambahkan untuk graph tak berarah
    }

    print_adjacency_matrix(&g);

    return 0;
}
```

Nomor 1

```
g->adjacency_matrix[i][j] = 0;
```

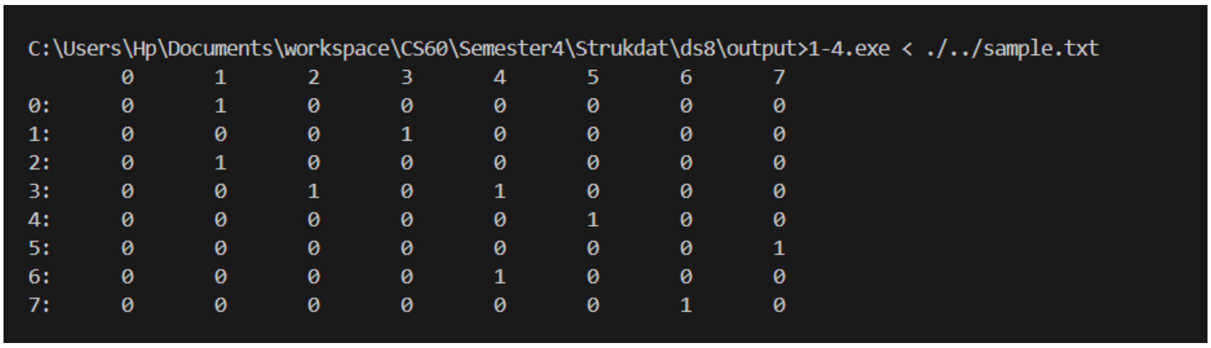
Nomor 2

```
g->adjacency_matrix[i][j] = -1;
```

Nomor 3

```
g.adjacency_matrix[a][b] = 1;
```

Input/Output



```
C:\Users\Hp\Documents\workspace\CS60\Semester4\Strukdat\ds8\output>1-4.exe < ./../sample.txt
        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
```

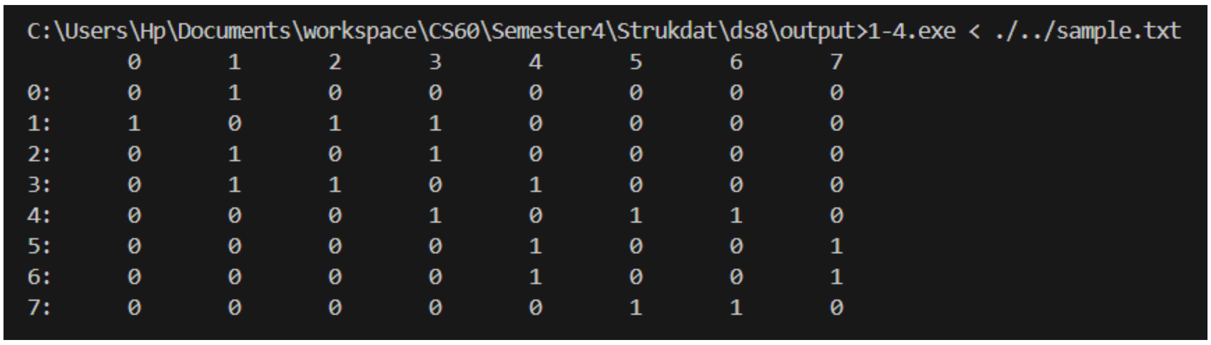
Nomor 4

```
  g.adjacency_matrix[b][a] = 1;
```

```
#include <stdio.h>
#define MAXNUM_VERTICES 100

typedef struct {
    int n_vertices;
    int n_edges;
    int adjacency_matrix[MAXNUM_VERTICES][MAXNUM_VERTICES];
} Graph;

void init_graph(Graph *g, int n_vertices, int n_edges) {
    int i, j;

    g->n_vertices = n_vertices;
    g->n_edges = n_edges;

    for(i = 0; i < MAXNUM_VERTICES; i++) {

        for(j = 0; j < MAXNUM_VERTICES; j++) {
            if(i < n_vertices && j < n_vertices) {
                g->adjacency_matrix[i][j] = 0;
            } else {
                g->adjacency_matrix[i][j] = -1;
            }
        }
    }
}

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_vertices = 0;
    int n_edges = 0;
    int i;
    scanf("%d %d", &n_vertices, &n_edges);

    Graph g;
    init_graph(&g, n_vertices, n_edges);

    for(i = 0; i < n_edges; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        g.adjacency_matrix[a][b] = 1;
        g.adjacency_matrix[b][a] = 1;  // tambahkan untuk graph tak berarah
    }

    print_adjacency_matrix(&g);

    return 0;
}
```

Input/Output



```
C:\Users\Hp\Documents\workspace\CS60\Semester4\Strukdat\ds8\output>1-4.exe < ./../sample.txt
        0       1       2       3       4       5       6       7
0:      0       1       0       0       0       0       0       0
1:      1       0       1       1       0       0       0       0
2:      0       1       0       1       0       0       0       0
3:      0       1       1       0       1       0       0       0
4:      0       0       0       1       0       1       1       0
5:      0       0       0       0       1       0       0       1
6:      0       0       0       0       1       0       0       1
7:      0       0       0       0       0       1       1       0
```

Nomor 5-6

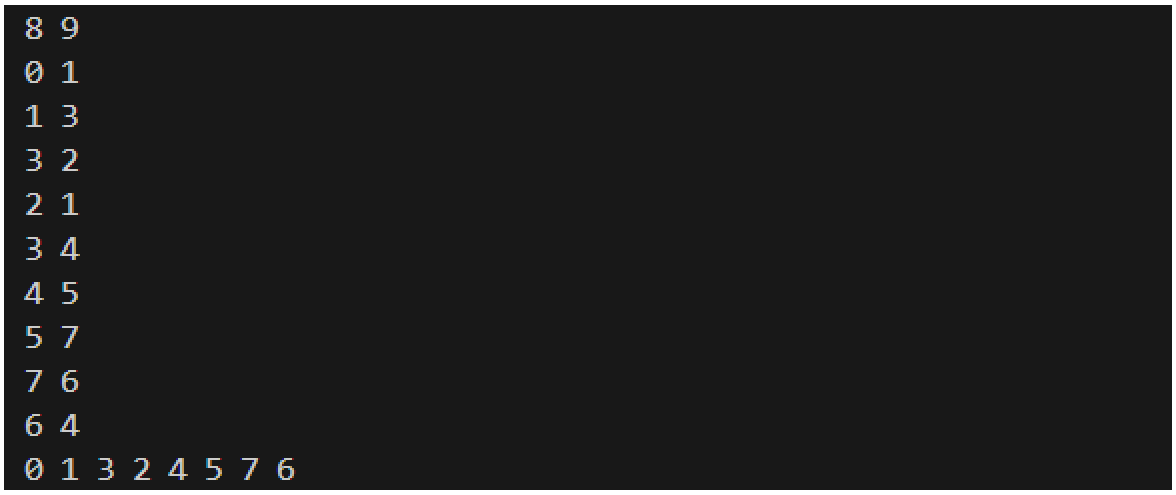
```c
#include <stdio.h>
#define MAXNUM_VERTICES 100

typedef enum {WHITE, GRAY , BLACK} Color;

typedef struct {
    int n_vertices;
    int n_edges;
    int adjacency_matrix[MAXNUM_VERTICES][MAXNUM_VERTICES];
} Graph;
```

```c
void DFS_visit(Graph *g, Color *vertex_colors, int v) {

    int i;

    printf("%d ", v);

    vertex_colors[v] = GRAY;

    for(i = 0; i < g->n_vertices; i++) {
        if(g->adjacency_matrix[v][i] == 1 && vertex_colors[i] == WHITE) {
            DFS_visit(g, vertex_colors, i);
        }
    }
    vertex_colors[v] = BLACK;
}

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");

}

int main() {

    Graph g;
    int n_vertices, n_edges, i, j;
    int a, b;

    scanf("%d %d", &n_vertices, &n_edges);
    g.n_vertices = n_vertices;
    g.n_edges = n_edges;

    for(i = 0; i < n_vertices; i++) {
        for(j = 0; j < n_vertices; j++) {
            g.adjacency_matrix[i][j] = 0;
        }
    }

    for(i = 0; i < MAXNUM_VERTICES; i++) {
        for(j = 0; j < MAXNUM_VERTICES; j++) {
            g.adjacency_matrix[i][j] = -1;
        }
    }

    for(i = 0; i < n_edges; i++) {
        scanf("%d %d", &a, &b);
        g.adjacency_matrix[a][b] = 1;
    }

    DFS(&g);
    printf("\n");

    return 0;
}
```

Nomor 5

```c
    for(i = 0; i < g->n_vertices; i++) {
        if(g->adjacency_matrix[v][i] == 1 && vertex_colors[i] == WHITE) {
            DFS_visit(g, vertex_colors, i);
        }
    }
```

Nomor 6 (Input/Output)



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