Lab12

Evaluation criteria

Category	Evaluation	
p12	100	
Total	100	

- *Use GCC **11** version*
- *No score will be given if the gcc version is different.*

Lab12 – Topological sorting

- The deadline for lab12 submission is May 31 at 11:59 PM.

- Folder name : lab12
- code name: p12.c
- Each code will be tested by 5 different input files.
- 20 score for each input, if you don’t get the answer you get 0 score.

Lab12 – Topological sorting

Queue *CreateQueue(int X)

- Create a new queue with the size of X.

void Enqueue(Queue *Q, int item)

- A new element at the end of the element in the queue.

int Dequeue(Queue *Q)

- The element in the front.

Lab12 – Topological sorting

Graph *CreateGraph(int X)

- Create vertices.
- Create adjacency matrix.
- All the input nodes will be positive numbers.

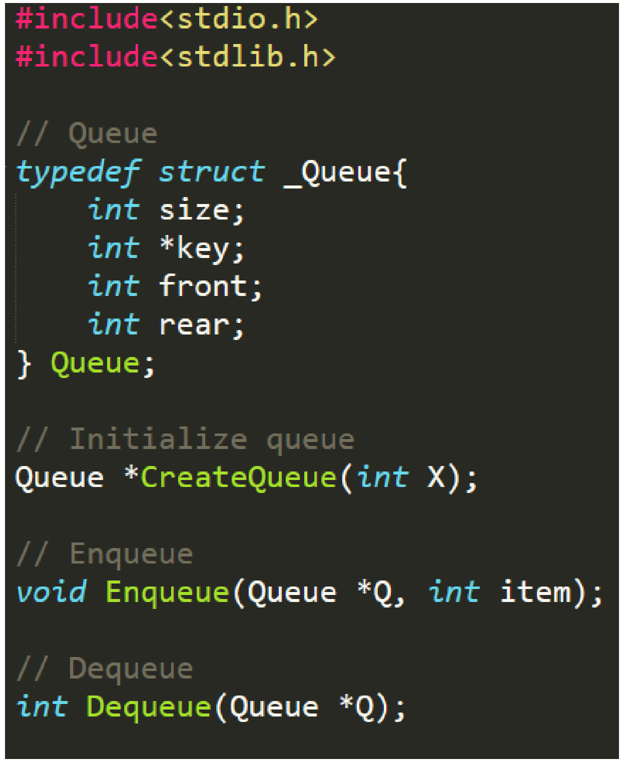
void InsertEdge(Graph *G, int u, int v)

- Insert a edge (u->v).

void Topsort(Graph *G)

- Print the graph by topological sort.

Lab12 – Topological sorting



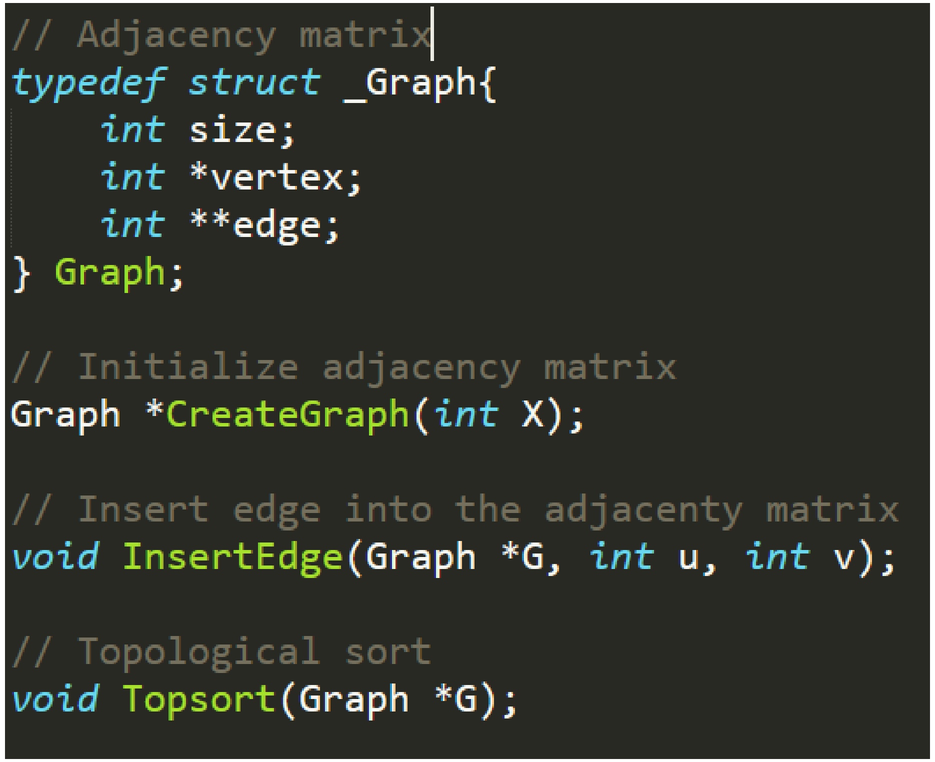
```c
#include<stdio.h>
#include<stdlib.h>

// Queue
typedef struct _Queue{
    int size;
    int *key;
    int front;
    int rear;
} Queue;

// Initialize queue
Queue *CreateQueue(int X);

// Enqueue
void Enqueue(Queue *Q, int item);

// Dequeue
int Dequeue(Queue *Q);
```

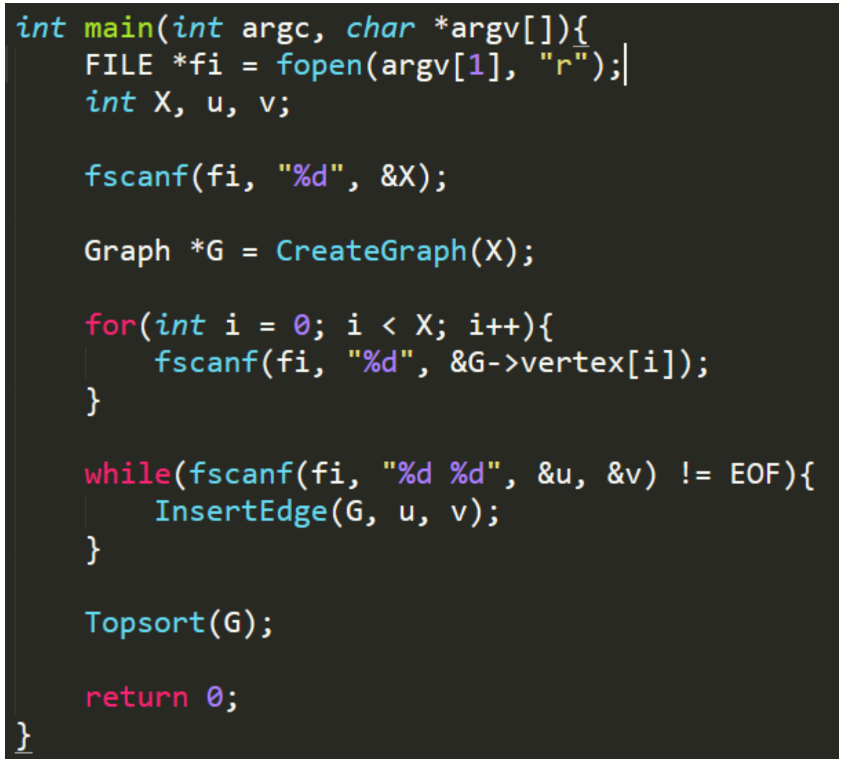


```c
// Adjacency matrix
typedef struct _Graph{
    int size;
    int *vertex;
    int **edge;
} Graph;

// Initialize adjacency matrix
Graph *CreateGraph(int X);

// Insert edge into the adjacenty matrix
void InsertEdge(Graph *G, int u, int v);

// Topological sort
void Topsort(Graph *G);
```



```c
int main(int argc, char *argv[]){
    FILE *fi = fopen(argv[1], "r");
    int X, u, v;

    fscanf(fi, "%d", &X);

    Graph *G = CreateGraph(X);

    for(int i = 0; i < X; i++){
        fscanf(fi, "%d", &G->vertex[i]);
    }

    while(fscanf(fi, "%d %d", &u, &v) != EOF){
        InsertEdge(G, u, v);
    }

    Topsort(G);

    return 0;
}
```

Lab12 – Topological sorting

Graph *CreateGraph(int X)

void InsertEdge(Graph *G, int u, int v)

- Ex) input.txt

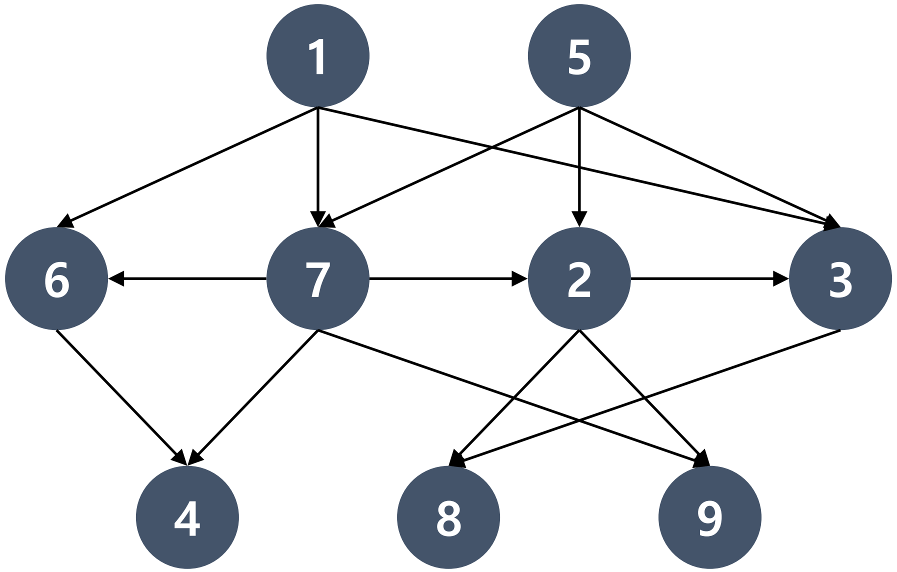
```
5
5 4 1 2 3
5 3
4 2
3 1
```

Vertices

	5	4	1	2	3
5	0	0	0	0	1
4	0	0	0	1	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	1	0	0

Adjacency Matrix

Lab12 – Topological sorting



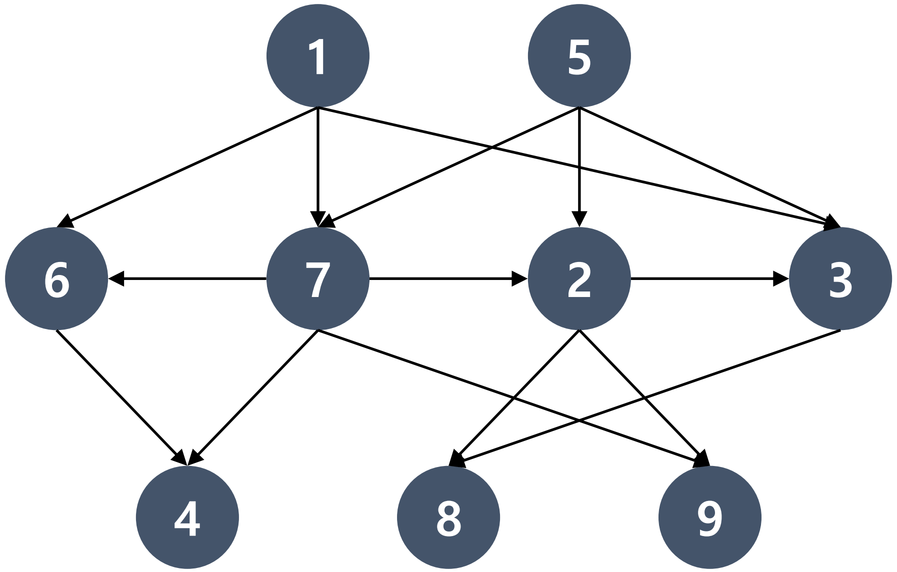
1	5	6	7	2	3	4	8	9

Vertices

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

Adjacency
Matrix

Lab12 – Topological sorting



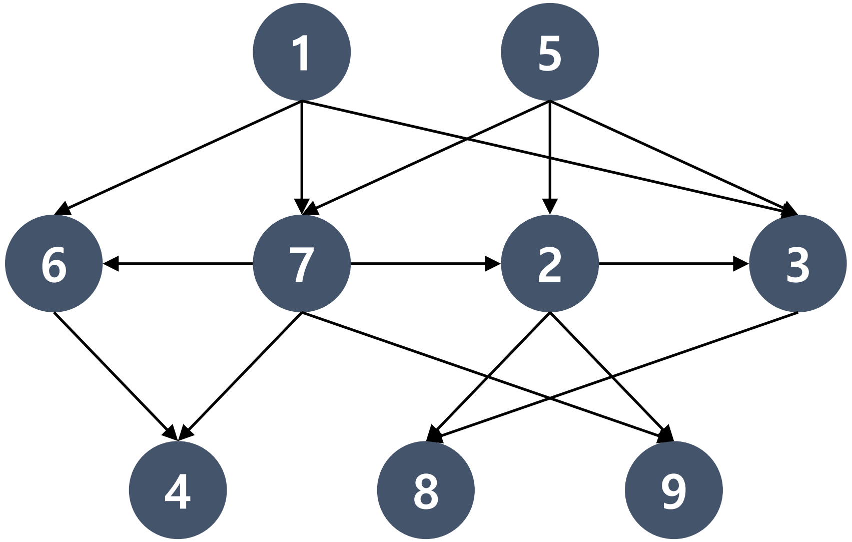
Vertices

Adjacency Matrix

Indegree

Vertices	1	5	6	7	2	3	4	8	9
1	■	0	1	1	0	1	0	0	0
5	0	■	0	1	1	1	0	0	0
6	0	0	■	0	0	0	1	0	0
7	0	0	1	■	1	0	1	0	1
2	0	0	0	0	■	1	0	1	1
3	0	0	0	0	0	■	0	1	0
4	0	0	0	0	0	0	■	0	0
8	0	0	0	0	0	0	0	■	0
9	0	0	0	0	0	0	0	0	■
Indegree	0	0	2	2	2	3	2	2	2

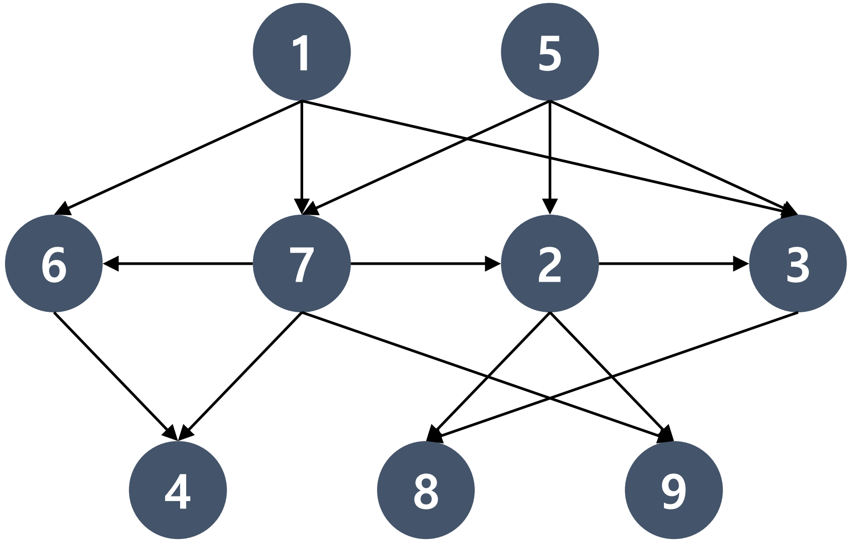
Lab12 – Topological sorting



1	5	6	7	2	3	4	8	9	Vertices
0	0	2	2	2	3	2	2	2	Indegree

Q

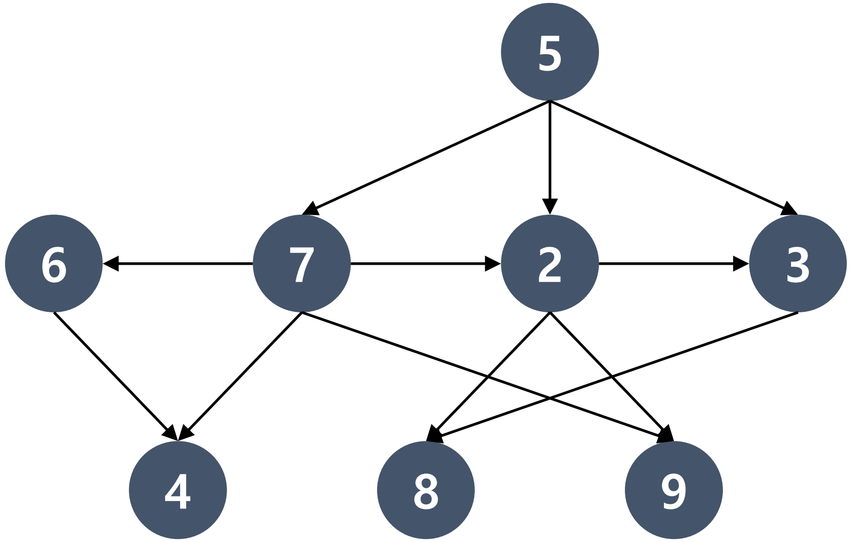
Lab12 – Topological sorting



1	5	6	7	2	3	4	8	9	Vertices
0	0	2	2	2	3	2	2	2	Indegree

1 5	Q

Lab12 – Topological sorting



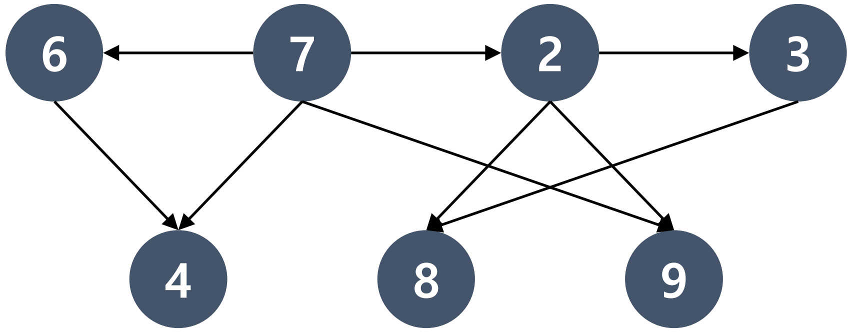
1	5	6	7	2	3	4	8	9	Vertices
0	0	1	1	2	2	2	2	2	Indegree

5

Q

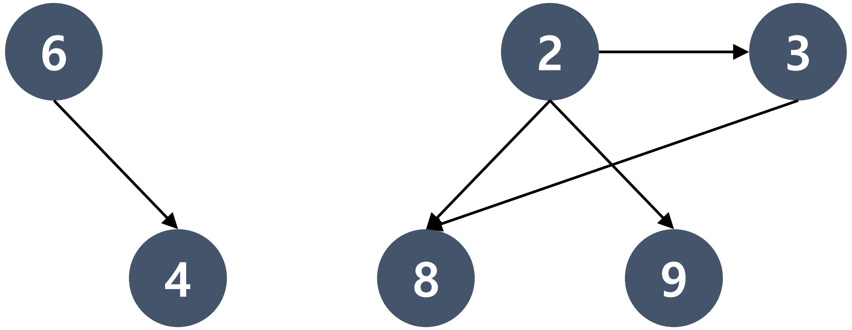
Lab12 – Topological sorting

1	5	6	7	2	3	4	8	9	Vertices
0	0	1	0	1	1	2	2	2	Indegree



7	Q

Lab12 – Topological sorting

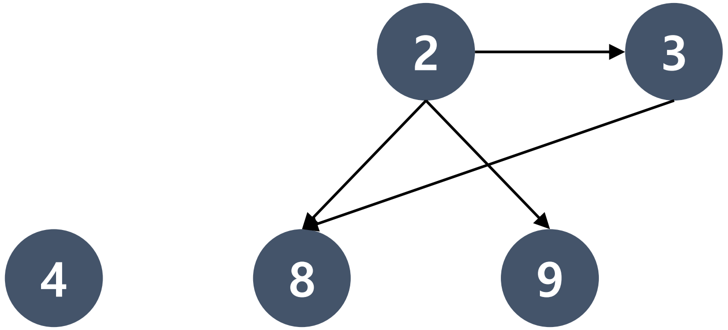


1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	1	1	2	1	Indegree

6 2

Q

Lab12 – Topological sorting

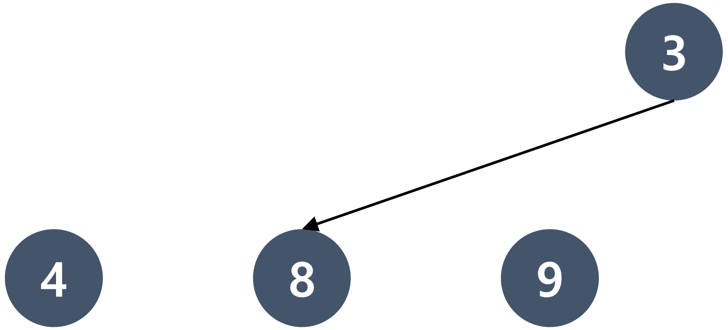


1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	1	0	2	1	Indegree

2 4

Q

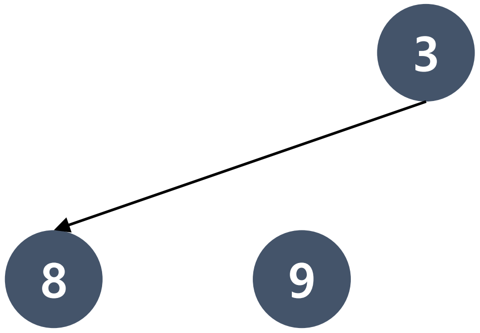
Lab12 – Topological sorting



1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	0	0	1	0	Indegree

4 3 9	Q

Lab12 – Topological sorting



1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	0	0	1	0	Indegree

3 9	Q

Lab12 – Topological sorting

1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	0	0	0	0	Indegree

9 8	Q

Lab12 – Topological sorting

1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	0	0	0	0	Indegree

8 Q

Lab12 – Topological sorting

1	5	6	7	2	3	4	8	9	Vertices
0	0	0	0	0	0	0	0	0	Indegree

Q

Lab12 – Topological sorting

- Input1.txt

```
5
1 2 3 4 5
1 2
2 3
3 4
4 5
```

- Out

```
1 2 3 4 5
```

- Input2.txt

```
6
11 13 15 17 19 21
11 13
11 15
13 17
13 19
15 17
15 19
17 19
```

- Out

```
11 21 13 15 17 19
```

- Input3.txt

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

- Out

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

- Input file
  - 1 : the number of vertices
  - 2 : vertex key (not index)
  - 3 ~ last : edge (a -> b)
- Output
  - Topological sorting result