CSC 2720: Data Structures
Lab 12

Instructor: Shiraj Pokharel

Due : @ 11:00 PM ET , Saturday 11/13.
Late Submission (with a 25% penalty) deadline : 11:00 PM ET, Sunday 11/14

Answer the below questions.
You may use whatever IDEs / editors you like, but you must submit your responses on iCollege as .java files.
Failure to comply with this simple requirement will result in a score of Zero. Please, be careful not to be assigned a Zero score this way.

Few Rules to be followed, else will receive a score of ZERO

(1) Your submissions will work exactly as required.

(2) Your files shall not be incomplete or worse corrupted such that the file does not compile at all. Make sure you submit a file that compiles.

(3) Your submission will show an output. Should you recive a Zero for no output shown do not bother to email me with ”but the logic is perfect” !

Note that your program's output must **exactly** match the specs(design , style) given here for each problem to pass the instructor's test cases .
Design refers to how well your code is written (i.e. is it clear, efficient, and elegant), while *Style* refers to the readability of your code (commented, correct indentation, good variable names).

PROBLEM STATEMENT :
In today's Lab we will explore a specific way to perform a Depth First Search (DFS) of a given Graph [Ref : Figure 1].

You will implement the traversal using one of the two ways stated below :

[1] With Recursion.

[2] Iteratively by using an explicit Stack.

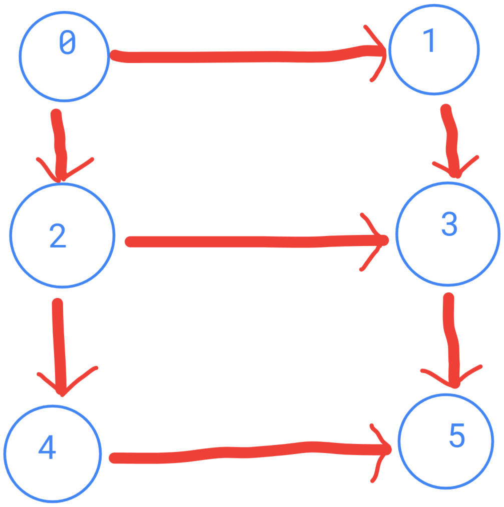

Figure 1: Graph for Traversal

```
/* Class representing a directed graph using adjacency lists */

static class Graph
{
   int V; //Number of Vertices

   LinkedList<Integer>[] adj; // adjacency lists

   //Constructor
   Graph(int V)
   {
      this.V = V;
      adj = new LinkedList[V];

      for (int i = 0; i < adj.length; i++)
         adj[i] = new LinkedList<Integer>();

   }

   //To add an edge to graph
```

```
void addEdge(int v, int w)
{
   adj[v].add(w); // Add w to the list of v.
}
```

The edges of the Graph is given to you.

```
g.addEdge(0, 1);
g.addEdge(0, 2);
g.addEdge(2, 3);
g.addEdge(2, 4);
g.addEdge(4, 5);
g.addEdge(1, 3);
g.addEdge(3, 5);
```

Your code will need to return the traversal of the nodes in DFS order, where the traversal starts from Node/Vertex 0.

When you follow the traversal process as specified - the complexity of the solution will be linear as shown below.

Time Complexity: $O(V + E)$, where V is the number of Vertices and E is the number of Edges respectively.
Space Complexity: $O(V)$
The linear space complexity would come from the recursion (AKA ”recursion stack”) you employ to traverse the Graph. If you solve the problem without recursion (using an explicit Stack), then the mentioned space complexity is obvious.
Submissions that don’t meet the linear Time and Space complexities will only receive 50% credit.

Very Very Important :

(1) Your code should be well commented which explains all the steps you are performing to solve the problem. A submission without code comments will immediately be deducted 15 points !

(2) As a comment in your code, please write your test-cases on how you would test your solution assumptions and hence your code.
A submission without test cases (as comments) will immediately be deducted 15 points ! Please Remember : Although, written as comments - You will address your test cases in the form of code and not prose :)