DATA STRUCTURES

BATCH – A & B (COMBINED LAB ASSIGNMENT)
[THURSDAY MARCH 23, 2017: 2:00 PM – 5:00 PM]

ASSIGNMENTS – 10 CODE: `assign10`

INSTRUCTIONS: [Total Marks: 25]

i) Read all assignments and each problem has to be answered in the same c file.
ii) Create a .c file following the file name convention: `abc-assign10.c`
Where `abc` is your roll number and `assign10` is the assignment code
iii) Strictly follow the file name convention and do not use `scanf()`

PROBLEMS: (Graphs – Basics and Adjacency Matrix)

1) **[Marks: 7 marks]**

Define a `GRAPH` - an undirected, weighted graph G = (V, E) as follows:

V = $\{v_i\}$ for all i such that $1 \leq v_i \leq n$ ($n = 20$) and $|V| = n$
E = $\{e_{ij}\}$ for all i and j such that $1 \leq \text{weight}(e_{ij}) \leq 5$
Here e_{ij} is the edge connecting the nodes v_i and v_j with specific weight in [1, 5]

```
GRAPH *buildUWGraph (int n);
```

a) You may represent the graph G as an adjacency matrix of size $n \times n$ by generating random values for set of vertices and edges with randomly assigned weights.

b) You have to allocate memory dynamically to generate this matrix and then store the nodes and edges in the matrix.

2) **[Marks: marks]**
Using the above adjacency matrix, do the following:

a) **[Marks: 3 marks]**
Find the degree of each of `n` nodes in the graph G

```
int nodeDegree(GRAPH *graph, int n);
```

This function should count the number of edges connected to each node in G.

b) **[Marks: 2 marks]**
Write a function to identify the paths of length k (= 3) and print the same

```
void printPathsOfLengthK(GRAPH *graph, int k);
```

c) **[Marks: 6 marks]**
Write a function to perform depth first search in the graph and print adjacency matrix of this traversal.

```
void performTraversalDFS (GRAPH *graph);
```

You may write separately additional functions, if needed, to perform Depth First Search.

d) **[Marks: 3 marks]**

Write a function to find the longest path in the given graph.

```
void findLongestPath(GRAPH *graph);
```

Print the nodes on the longest path with their degrees.

e) **[Marks: 4 marks]**
Write a function to delete all nodes of specific degree `k` and print the resulting graph.

```
void deleteKdegreeVertices(GRAPH *graph, int k);
```