DATA STRUCTURES

BATCH – A & B (COMBINED LAB ASSIGNMENT)
[THURSDAY MARCH 30, 2017: 2:00 PM – 5:00 PM]

ASSIGNMENTS – 11 CODE: `assign11`

INSTRUCTIONS: [Total Marks: 40]

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-assign11.c`
Where `abc` is your roll number and `assign11` is the assignment code
iii) Strictly follow the file name convention and do not use `scanf()`

iv) Remember to create additional functions, if necessary

PROBLEMS: (Graphs – Adjacency List, Spanning Trees, Shortest Paths / TRIE)

1) **[Marks: 10 marks]**

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

$V = \{v_i\}$ for all i such that $1 \le v_i \le n$ ($n = 30$) and $|V| = n$

$E = \{e_{ij}\}$ for all i and j such that $0 \le \text{weight}(e_{ij}) \le 1$

Here e_{ij} is the edge connecting the nodes v_i and v_j with specific weight in [0, 1]

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

a) You may represent the graph G as an adjacency list by generating random values for set of vertices and edges with randomly assigned weights.
b) You have to allocate memory dynamically to generate the adjacency list and then store the nodes and edges in this adjacency list.

2) **[Marks: 10 marks]**

Do the following tasks:

a) Construct a spanning tree of the above graph

```
GRAPH *getSpanningTree(GRAPH *graph);
```

b) Print the degree of each node in the spanning tree.
```
void printGraph(GRAPH *stree);
```

3) **[Marks: 10 marks]**

Write a function to find all shortest paths between any 2 vertices in a graph such that the total sum of the edges weights is less than or equal to 6.

```
void *findShortestPathsL6(GRAPH *graph, int source, int dest);
```

4) **[Marks: 10 marks]**

Just refer to the TRIE code given in the practice section:
https://www.2power3.com/rajendra/assignments/codes/sol-lc-trie.c

Load the names given in the text file – "`names-authors.txt`" - using TRIE data structure.

Each line in this file represents a name. Choose 200 names randomly from this text file.

Using the code given in the practice session, load these names in the TRIE data structure.

Now your task is to find the common prefix in these names in such a way that the size of the common prefix is greater than `k` (initially assume `k` to be 3 and work out for any value of `k`).

```
Void findPrefixofSizeK(TRIE *trie, int k);
```