SGSITS
Dept. of Computer Engineering
CO24007: Data Structure
Lab Assignment 5B
Graphs

Submission Date: **Monday 18th September, 2017 @23:59**
Demo Date: Monday 18th September, 2017 to Monday 25th September, 2017
Late Submission: Not allowed
No copying allowed. If found then students involved in copying will fail in this course.

Note:

1. Use graph in 'facebook_combined.txt' file for all questions in this assignment.
2. If data is entered through standard input, then user first enters two space separated numbers denoting number of vertices (V) and number of edges (E). On the next E lines, there will be two space separated numbers denoting the two vertices V_1 and V_2, which are connected.

1. **Simple Graph Operations**

   (a) **void getVertexDegree(int vertexIndex)**: This function prints degree of the vertex *vertexIndex*.

   (b) **void breadthFirstSearch(int vertexIndex)**: This function starts breadth first search with vertex *vertexIndex* and prints the elements in breadth first search order.

   (c) **void depthFirstSearch(int vertexIndex)**: This function starts depth first search with vertex *vertexIndex* and prints the elements in depth first search order.

2. **Shortest Path**

(a) Write a function **void printShortestPath(int V1, int V2)**, it accepts two arguments and length of the shortest path from V1 to V2, and prints the path from V1 to V2.

(b) Write a function **void printAllShortestPaths()**, which prints shortest path and its length for each pair of nodes. Assume the graph is undirected.

3. **Clique**
A clique is a complete graph, i.e. every node is connected to every other node. Write a function **int isGraphAClique()** which returns 1 if the graph froms a clique, 0 otherwise.

4. **Betweenness centrality** Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. Write a function **void betweennessCentrality(int vertexIndex)** that return betweenness centrality of a vertex $vertedIndex$. (Hint: use your solution from shortest path problem).

5. **Page Rank Algorithm** Page Rank is the ranking algorithm that was developed by Larry Page and Sergey Brin (founders of Google Inc.), which is used in searching information on the Internet.

Internet is can be considered as a large graph with different pages as vertices and links between them as edges. Following is earliest version of PageRank algorithm.

$$PR_{t+1}(u) = (1-d) + d \sum_{v \epsilon N(u)} \frac{PR_t(v)}{D(v)}$$

where, $PR_t(v)$ = pagerank of vertex v after t^{th} iteration
d= damping factor (typically 0.85)
$N(u)$= set of neighbours of u
$D(v)$= degree of v

Implement iterative page rank algorithm using above formula. In each iteration compute pagerank for every node in the graph using pageranks computed in previous iteration. Initialize pageranks for each vertex to be 0. Your program should stop when the accuracy of pageranks is high enough. For this exercise, your program should calculate error in pageranks for each vertex, if the error is smaller than or equal to 0.01 for every vertex, it is time to stop iterating.