Celal Bayar University Computer Engineering Department
CSE 2105 Data Structures

Project - III ***17 December 2014***

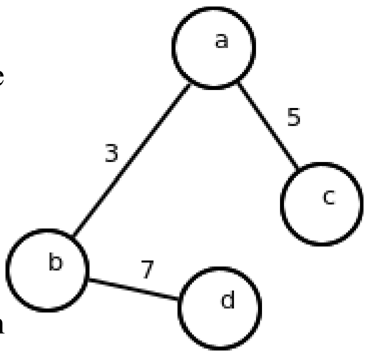
You should upload the homework to the ***Moodle*** *until* ***31 December 2014 23:55****. Don't forget to write your name and your number. You should upload in fully working program as a zip/rar file with all required files. You can ask your questions to me or* ***Res. Assist. Zeynep ÇİPİLOĞLU YILDIZ****.*

Assist. Prof. Dr. Tahir Emre KALAYCI

Project Specifications and Requirements

In this homework you need to develop a fully working graph data structures with operations explained below. The main objective of the homework is the development of efficient and convenient data structures for the program. *Instead of using Java Data Structures Libraries you should develop your own data structures or use the ones you developed in laboratory classes.* You should follow object oriented programming methodology.
A graph **G = (V, E)** consists of a set of vertexes **V**, and a set of edges (arcs) **E** between those vertexes. Edges in graphs can be directed or undirected, the difference being whether the relationship is mutual or one-sided. Edges can be weighted, the value of the weight represents some quantitative measure. Most often these weights are thought of as being the distance between the connected vertexes. A path is a sequence of edges that connects two vertexes. There may be many paths, or no paths, between two vertexes in a graph.

By using any representation that you want for the graphs (adjacency matrix, adjacency list, etc.), write the program which performs the following operations using the graph data structure that you developed. You should assume that the edges is undirected and weighted in the graph.



1. **Creating a graph by the user input:** For example to store the graph on the right in your program, user should enter the following input:

   >a(b,3)

   >a(c,5)

   >b(d,7)
2. **Printing the graph on the screen** using BFS and DFS traversal starting from a custom node (Example: output of the graph on the right starting from node **a** BFS will be **abcd** and DFS will be **abdc**)
3. **Shortest Path** you can use Dijkstra'a algorithm or any other algorithm.
4. **Minimum Spanning Tree** using Prim's or Kruskal's algorithm.
5. **Searching** for an element in the graph
6. **Exiting** from the program
7. *(Optional for bonus)* Saving the graph as **TXT** (using the graph input format) file at the exit
8. *(Optional for bonus)* Performing these operations by the help of GUI (Java Swing)

Operation between **2-5** can only be performed, after the graph is created by the user input.