Extra Credit.
Due date: Friday, December 8 2017 at 03:30pm

Every 10 points = 1% of the final grade

1. **10 points:** Write a function to sort random numbers entered by a user using **BST**. (Print your BST and the output of the sorted numbers.)
2. **10 points:** Write a function to make an **AVL tree**. To test, ask the user to insert random numbers, insert the numbers one by one and when required do single/double rotation(s), and then print your AVL tree.
3. **10 points:** Write a function to find a shortest path in a weighted graph (**Dijkstra**).
4. **10 points:** Write a function to find the MST on an undirected weighted graph (either **Prim** or **Kruskal**).
5. **40 points: Highly recommended**

In this program you are required to implement the BFS and DFS algorithms.

a. Request the user to determine the order (|V|) and size (|E|) of the graph.
b. Generate **|E|** random ones into the adjacency matrix/list (**Adj**) to make a random *directed* graph.
c. Print the resulting adjacency matrix/list.

Part A.

d. Request the user to determine the starting vertex (*u*) for **BFS** and **DFS_visit** algorithms
e. Call BFS(u, Adj) function to find the vertices reachable from vertex u and print the *shortest paths* and their *lengths/distances* from u to the reachable vertices.
f. Call DFS_visit(u,Adj) function to find the vertices reachable from vertex u
   - ✓ For each vertex print the *start/finish* time.

Part B.

Write a code to print the topological order of the vertices.

a. Run DFS(V,Adj) to check if the graph is DAG (directed acyclic graph):
   - ✓ Search for backward edges. If there are any, (the graph has a cycle.) print: “Cycle detected, topological sort is impossible”.

b. If the graph is DAG, (while running DFS):

- ✓ For each vertex v as it finishes, insert it onto the front of a linked list
- ✓ Print the topological order of vertices along with their finish/start