1. Write a program that can:
read a text file of graph information with each edge of the graph containing <outVertex, inVertex, weight> to create an *edge* Comparable object.
store the information for each *edge* in an **Adjacency Matris (Table), Adjacency List of Edges, Adjacency List with Valence (hash table by outVertex).** (note: the Adjacency Matrix/Table can use primitives for the weight instead of the *edge* object!)

Your program should then find the **edge of minimum weight** for each of the representations. Include a "count" of the comparisons needed for each method, and compare the results to "theoretical" counts of compares.

Then try using a Sorted Adjacency List (by weight) and Sorted "Hash Table" (by weight) to count the number of compares to find the **edge of minimum weight**.

2. Write a program that can:
read a text file of graph information with each edge of the graph containing <outVertex, inVertex, weight> to create an *edge* Comparable object.
store the information for each *edge* in an **Adjacency Matris (Table).** (note: you can use primitives in the Adjacency Matrix.)

Find the "shortest paths" between all pairs of vertices using:
All Pairs Shortest Path Algorithm (Floyd)
Dijkstra Shortest Path Algorithm – using each vertes as the Start vertex.
Include counters for the number of steps used by each algorithm and discuss the results.
Test the algorithm with a graph with all edges >0, and with a graph including some edges<0.
Discuss the results.