You will write a program that accepts as input a file (see below) with edges and weights of a digraph. The program will prompt the user for the path to the file. If no path is supplied, it will by default look for the file in path c:\\temp\\assignment6_input.txt

20 pts. The input file has the following information:

1.number of vertices

2. number of edges

3. the tuples "u v w" where u is the source vertex, v is the adjacent vertex, and w cost to get from u to v.

For example (not the comments, they are for your understanding):

```
9        // number of vertices
8        // number of edges
1 2 3    // u=1 v=2 w=3
2 5 1    // u=2 v=5 w=1
1 4 4    // u=1 v=4 w=4
5 6 3
6 7 2
6 8 6
8 9 7
4 6 1
```

50 pts. Your program will define a function getLongestPath(i) such that l(i) is the length of the longest/most expensive path that originates at vertex i. In the example above, if i=1, the longest path is 20 from node 1 to 9. NOTE: you do not supply the destination vertex, only the source vertex.

Hint: Let S(i) be the set of vertices that are adjacent from vertex i (i.e., the vertices j such that (i,j) is an edge of the graph). We obtain the following recurrence for l(i)

l(i) = 0 if S(i) is empty and l(i) = max{cost(i,j) + l(j) | j is in S(i)}

30 pts. Your program must also output the following (not the comments, they are for your benefit):

The input weighted digraph is:

Vertex 1 = [4 4, 2 3] // this means that vertex 4 is adjacent to vertex 1 with a weight of 4 and vertex 2
// is adjacent to vertex 1 with a weight of 3

Vertex 2 = [5 1] // this means that vertex 5 is adjacent to vertex 2 with a weight of 1

Vertex 3 = [] // this is an unconnected component, no other vertex reaches it

Vertex 4 = [6 1]

Vertex 5 = [6 3]

Vertex 6 = [8 6, 7 2]

Vertex 7 = [] // has no outbound edges.

Vertex 8 = [9 7]

Vertex 9 = [] // also has no outbound edges

The length of the longest path is: 20

The longest path is: 1 2 5 6 8 9