Project 3 Milestone 1

Wenjing Bao

I decide to divide the whole program into smaller functions to make it easier to debug, besides, I will also create a user-defined structure to store vertices.

Structure:

vert: it will contain two **int** called **x** and **y** to store its coordinates, another two **int** called **dist** and **from** for the distance and from where that we need to keep trace of when using Dijkstra's shortest path algorithm.

Functions:

main: this is the main function that call the following functions to do the trick.

readmap: it will read in the total number of vertices and edges first. Then use the number of edges to create a two-dimensional array (3 lines * number of edge) called **edges** to store the edges (one point at the first subarray, one on the second subarray) and their length (in the 3rd subarray) in. Then use the number of vertices as the limit of iterations to create a **for** loop that read a line of input file in each iteration and call **createvert** to create a **vert** structure. And store the pointers to all **vert** into an array called **vertices** based on the name of vertices given in the input file. Then run another for loop use number of edges to limit iterations, in each iteration, it will read a line from the input file in a certain way and store the two ends of the edge into **edges**, then return **vertices** and **edges**.

ms2: this is the function to complete milestone 2, it will run a for loop (number of edges as iteration limit). At the start of each iteration, it will print out the current iteration number to the output, then read through the first 2 lines of **edges** to find the current iteration number, when found one, it will print the number at the other line to the output. At the end of search, it will print the "\n".

createvert: it will allocate memory and reset a **vert**, then put the number given in the parameters into the structure and pass it back.

calc: this function will read one element from each of the first 2 lines of **edges** and calculate the length of the edge based on the coordinate in the corresponding **vert**.

readque: this will read in the number of query first, then read in the pair of vertices and call **dijk** below to get the result and write them in the output.

dijk: it will first reset all the **dist** and **from** in all **vert**s to -1, then run the Dijkstra's shortest path algorithm like what shows in the wiki. (I will keep the shortest route by use **from** to keep where the shortest route come to a certain point from)