ENGR 476 ENGINEERING Lab

HW - 3

Student name: Anish Kumaramangalam

Section: 01

Date: Mar. 29, 2016

ENGR 476 : computer Communications Networks
Lab: shortest path finder

Code:

```
////////////////////////////////////////////////////////////////////////////
//
// Title:            pathfinder.c
// Problem:   pathfinder finds the shortest path from start node to end node
//         with a list of intermediate nodes and total cost.
//
// Class:             ENGR 476
// Date:              03/29/2016
// Author:          Anish Kumaramagalam
//
////////////////////////////////////////////////////////////////////////////


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

char*  addstr  (char* str1, char* str2);
int extract();
int getNodes(int i);

/*  global var  */
const int limit = 30;// WARNING! this program will only host limit number line in map
char data[limit][50];   // stores file locally
char nodes[limit][50];
int distance=0;    // counts people who pass
int path[limit];
char temp[2];
int Lpath = 0;
char toNode, findNode, input[1];

/*  main  */
int main()  {

   /*  initializing    */
   int c = 0, pathsize = 2;
   int r = extract();
   const int n = getNodes(r);

   int distance[n][n]; // CONTAINS POINT TO POINT DISTANCES
```

```
int point[n][n];// CONTAINS POINTS

printf("\n");

/*need to add -code- to get input and to node*/

printf("enter from node : \n");
scanf("%s",input);
toNode= input[0];
path[0]= toNode;
printf("enter destination node : \n");
scanf("%s",input);
findNode=input[0];

/*  path initiation */
path[0] = ((int)toNode)-65;
path[1] = ((int)findNode)-65;

printf("\n");

// populate point and distance mat
for (int i = 0; i < n; ++i) {
   for (int j = 0; j < n; ++j) {
      point[i][j] = -1; // really really small number
      distance[i][j] = 100000000; // really really big number to be like infinity
   }
}

for (int i = 0; i < r; ++i) { // adds all point to point distances to dist mat
   distance[((int)data[i][0])-65][((int)data[i][1])-65] = ((int)data[i][3])-48;
}

/*  increasing resolution of point and distance mat*/
for (int k = 0; k < n; ++k) {
   for (int i = 0; i < n; ++i) {
      for (int j = 0; j < n; ++j) {
         if ((distance[i][k]+distance[k][j]) < distance[i][j])   {
            point[i][j] = k;
            distance[i][j] = (distance[i][k]+distance[k][j]);
         }
      }
   }
}
```

```
/*  tries to find the nearest distance to findNode   */
while(1)    {

   // checks if their is point in between path[c+0] and path[c+1]
   if (point[path[c+0]][path[c+1]] > -1)   {

      for (int i = pathsize - 1; i >= c+1; i--)
         path[i+1] = path[i];

      path[c+1] = point[path[0+c]][path[1+c]];
      pathsize++;

   } else  {
      /*
      * - if start point and end point have no point in between Beaks
      * - adds distance to path distance
      * - moves c pinter to next two point to check if their is a point
      *      in between them
      */

      Lpath += distance[path[0+c]][path[1+c]];
      if(pathsize == 2)    {
         break;
      }
      c++;
   }
   /*
   for (int i = 0; i < limit; ++i) {
      printf("%d", path[i]);

   }
   printf("\n");
   */

   if(pathsize == limit)    { // error when path size exceeds  path limit
      printf("error exceeded path limit\n");
      break;
   }

   if(path[c+1] == 0) {
      break;
   }
```

```
    }

    /*  prints outputs  */
    printf("path :");
    for (int i = 0; i < pathsize; ++i) {

        printf("%c/", (char)(path[i]+65));
    }
    printf("\n\npath length : %d\n\n", Lpath);

    /*
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            printf("%d", point[i][j]);
        }
        printf("\n");
    }
    printf("\n");

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            //printf("%d", point[i][j]);
            printf("%d", distance[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    */

    return 0;
}   // end main

char*  addstr  (char* str1, char* str2)  {

    char * str3 = (char *) malloc(1 + strlen(str1)+ strlen(str2) );
    strcpy(str3, str1);
    strcat(str3, str2);

    return str3;
}

int extract()   {
    int i = 0;
```

```
   /* read file */
   char line [50]; // temp
   char file_name[] = "map.txt"; // file name
   FILE* fp = fopen(file_name,"r"); // opens file read mode

   while (fgets(line, sizeof(line), fp)) {

      strcpy(data[i], line);
      i++;
   }

   fclose(fp); // closing file
   return i;
}

int getNodes(int i)   { // gets a list of nodes in the map
   int k =1;
   toNode = data[0][0];
   nodes[0][0] = toNode;
   nodes[0][1] =' ';
   nodes[0][2] ='x';

   for (int j = 0; j < i; ++j) { // reads all node distances

      if(data[j][0] != toNode) {// finds the nodes
         toNode = data[j][0];
         nodes[k][0] = toNode;
         nodes[k][1] =' ';
         nodes[k][2] ='x';
         k++;
      }

   }

   return k;
}
```

Output:

```
Anishs-MacBook-Pro:lab-3 anishkumaramangalam$ gcc pathfinder.c -o pathfinder
Anishs-MacBook-Pro:lab-3 anishkumaramangalam$ ./pathfinder

enter from node :
A
enter destination node :
B

path :A/B/

path length : 3

Anishs-MacBook-Pro:lab-3 anishkumaramangalam$ ./pathfinder

enter from node :
A
enter destination node :
C

path :A/C/

path length : 2

Anishs-MacBook-Pro:lab-3 anishkumaramangalam$ ./pathfinder

enter from node :
A
enter destination node :
D

path :A/B/D/

path length : 5

Anishs-MacBook-Pro:lab-3 anishkumaramangalam$
```