**Student Number: 202135760**

**Student Name: 김현지**

**Problem 9**

* **Screenshots**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* **Code**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define TRUE 1

#define FALSE 0

#define INF 2147483647//infinte

#define MAXINPUT 30 //maximum number of inputs

typedef struct Vertex {

float x;

float y;

};

int numOfInput; //number of inputs

Vertex vertex[MAXINPUT];

float weight[MAXINPUT][MAXINPUT] = { INF, }; //weight[v][w] : cost from v to w

int visit[30] = { FALSE, }; //visit[v] == true : already visited | init false

float getMinVertex(int\* minVertexvIdx, int\* minVertexwIdx ) {//return minweight

//init

float minVertexWeight = INF; //min cost

for (int wIdx = 0; wIdx < numOfInput; wIdx++) {

if (!visit[wIdx]) {

\*minVertexwIdx = wIdx; //Index of unvisited point

break;

}

}

for (int vIdx = 0; vIdx < numOfInput; vIdx++) {

for (int wIdx = 0; wIdx < numOfInput; wIdx++) {

if (visit[vIdx] && !visit[wIdx] && weight[vIdx][wIdx] < minVertexWeight) {//v = point included in current graph, w = point not included in current graph

\*minVertexwIdx = wIdx;

\*minVertexvIdx = vIdx;

minVertexWeight = weight[vIdx][wIdx];

}

}

}

return minVertexWeight;

}

float prims() {

visit[0] = TRUE;

float minCost = 0;

int vIdx, wIdx;

for (int i = 0; i < numOfInput-1; i++) {

minCost +=getMinVertex(&vIdx, &wIdx); //get min vertex connected to the current graph

visit[wIdx] = TRUE; //add point w to the current graph

}

return minCost;

}

void main() {

float x, y;

int vIdx, wIdx;

int dataSetCount = 0;

//input

do {

printf("Enter num of data (0<n<30): ");

scanf("%d", &numOfInput);

} while (numOfInput <= 0 || numOfInput >= 30);

printf("Enter data : x y\n");

for (int i = 0; i < numOfInput; i++) {

printf("[%d] ", i);

scanf("%f %f", &x, &y);

vertex[i].x = x;

vertex[i].y = y;

}

//calculate cost

for (int vIdx = 0; vIdx < numOfInput; vIdx++) {

for (int wIdx = 0; wIdx < numOfInput; wIdx++) {

if (vIdx == wIdx)

continue;

weight[vIdx][wIdx] = sqrt(pow((vertex[vIdx].x - vertex[wIdx].x), 2) + pow((vertex[vIdx].y - vertex[wIdx].y), 2));

}

}

//result

printf("minCost = %.2f", prims());

}

**Problem 2**

* **Screenshots**
* **Code**