고급컴퓨터프로그래밍 기말 프로젝트 설명서

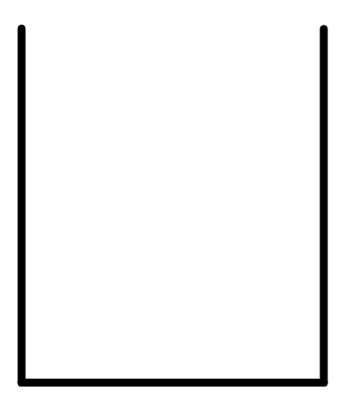
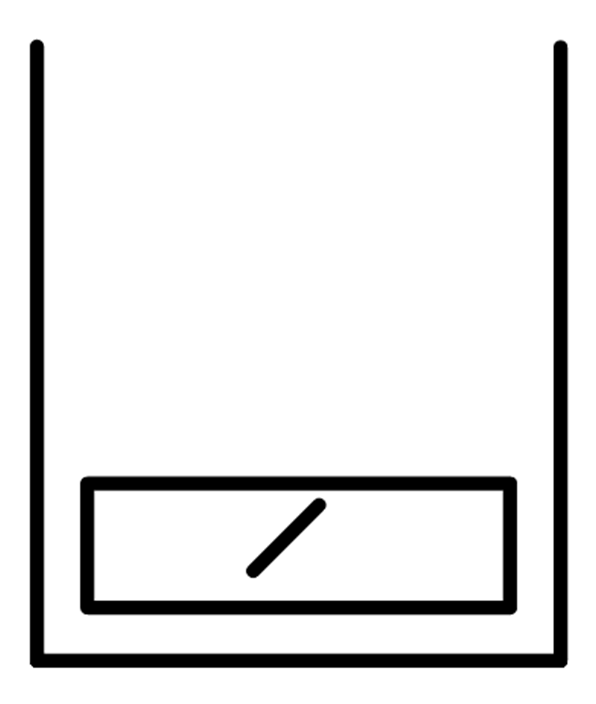
2020069047김태현

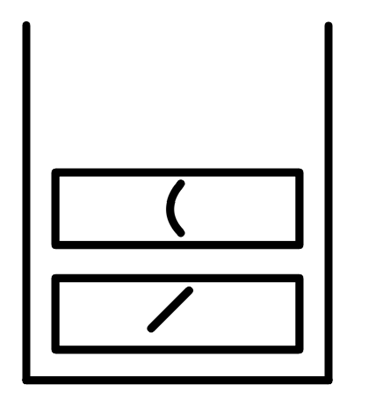
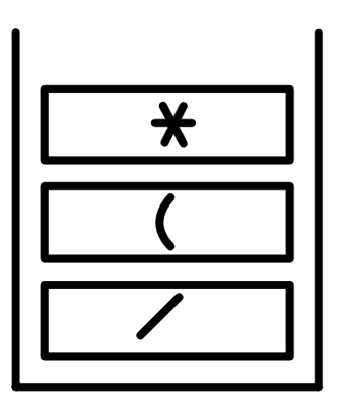
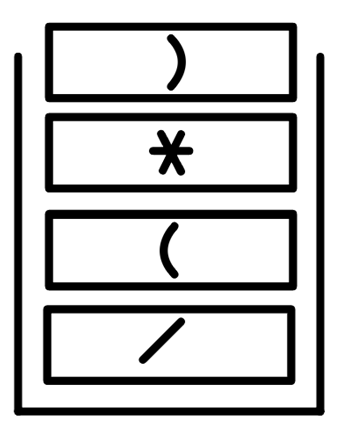
1. 프로그램 구성

저는 기본주제를 사용자의 키와 몸무게를 입력 받으면, BMI를 고려하고 그에 맞는 옷 사이즈를 추천해 준 뒤, 현재 위치에서 옷매장까지 가는 최단경로를 계산해주는 프로그램을 구상해주었습니다. 프로그램에서 키와 몸무게를 받고 계산할 때, 사용자는 중위연산자로 타이핑하지만 컴퓨터는 이를 후위연산자로 계산하여 값을 출력해줍니다. 따라서 저는 사용자가 BMI계산식 몸무게/(키\*키)을 입력하면 컴퓨터가 어떻게 입력 받는지 나타내 주기 위하여 stack을 사용하여 중위연산자를 입력 받으면 후위연산자로 출력하도록 프로그램을 구성하였습니다.

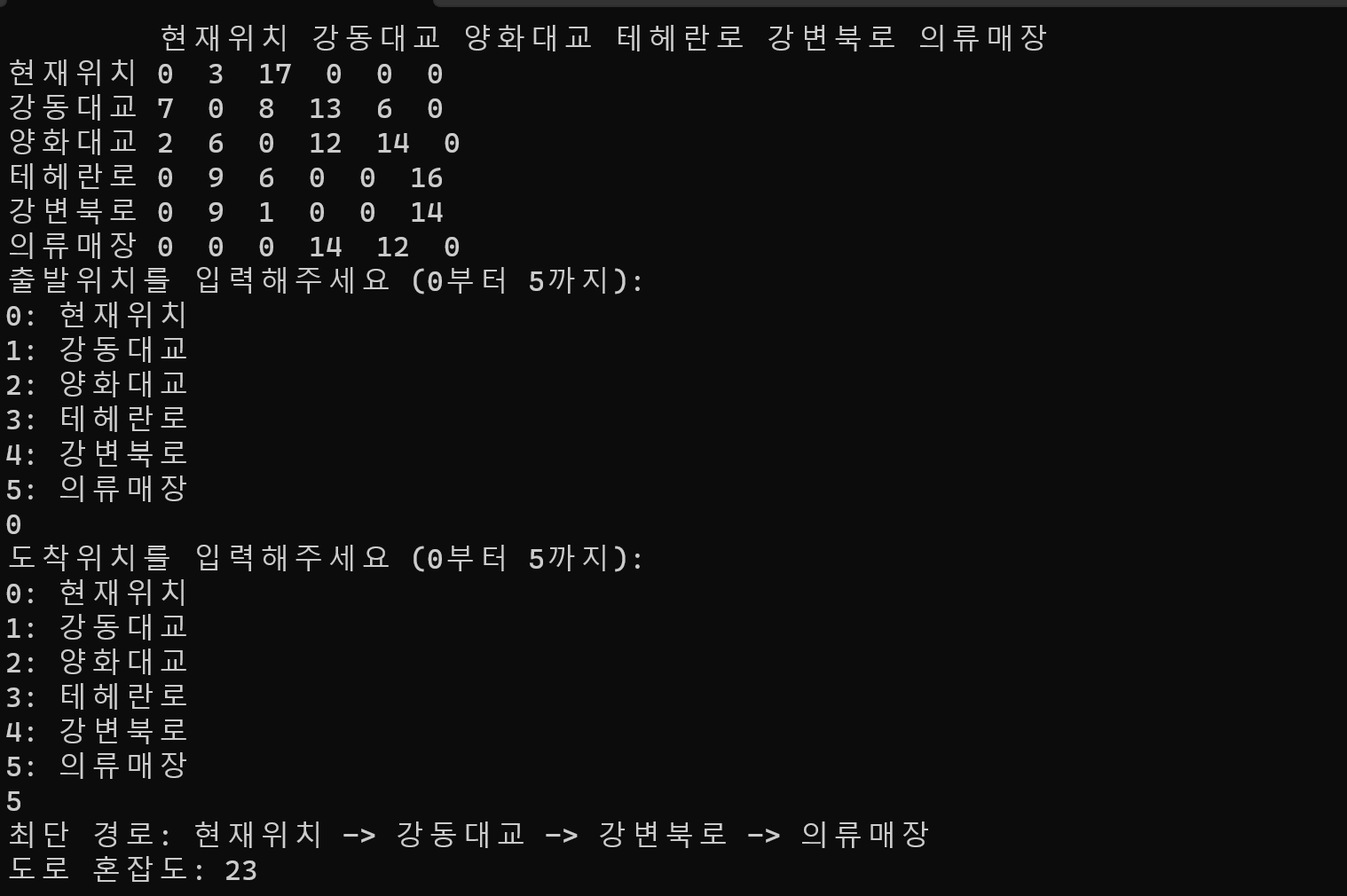
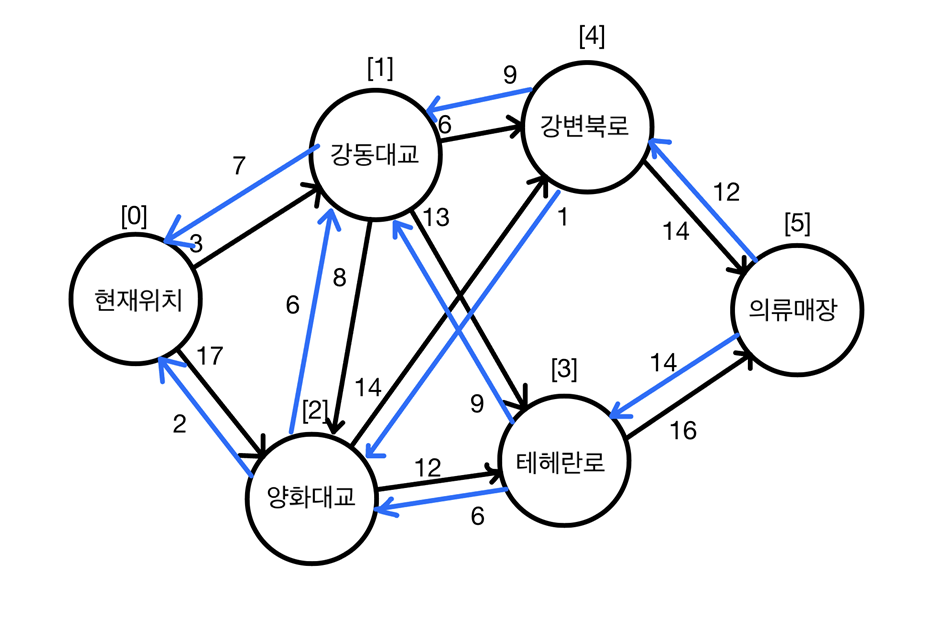
몸무게를 A,키를 B라고하면 중위표기법에서는 BMI계산식이 A/(B\*B)같이 표시될 것입니다

Stack을 이용해서 이를 변환하여 준다면 stack에는 연산자 값이 들어갈 것이고 피연산자는 출력으로 넘어가게 될 것입니다.

입력 중위연산자 A/(B\*B)  출력: → 입력:(B\*B)  출력:A

→입력\*B)  출력:AB →입력:)  출력ABB→ 입력:null  stack의 괄호연산자 완성 ‘)’ 와 ‘(‘ 가 출력될 때까지 pop , 마지막 나누기연산자도 POP 최종출력:ABB\*/

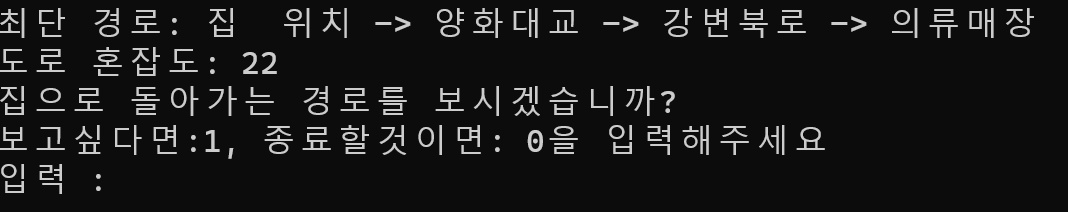
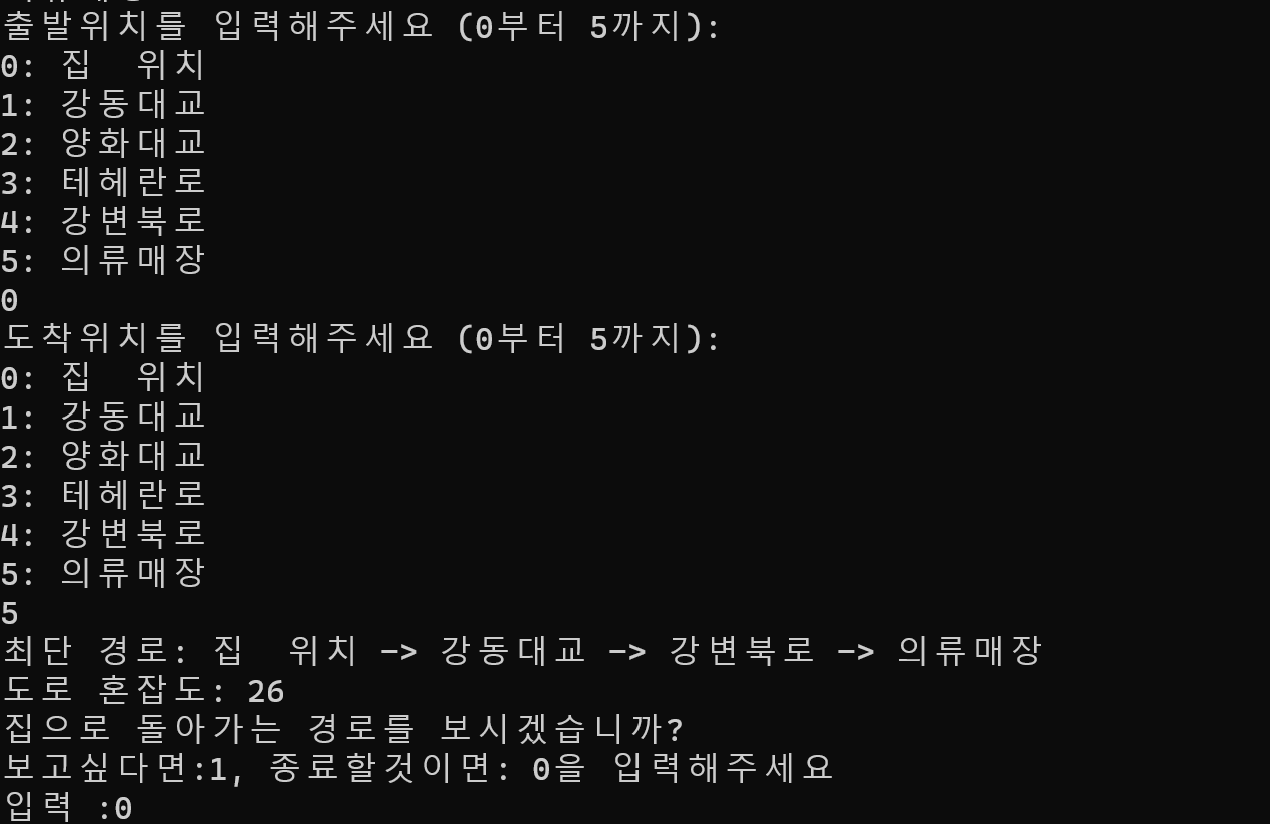
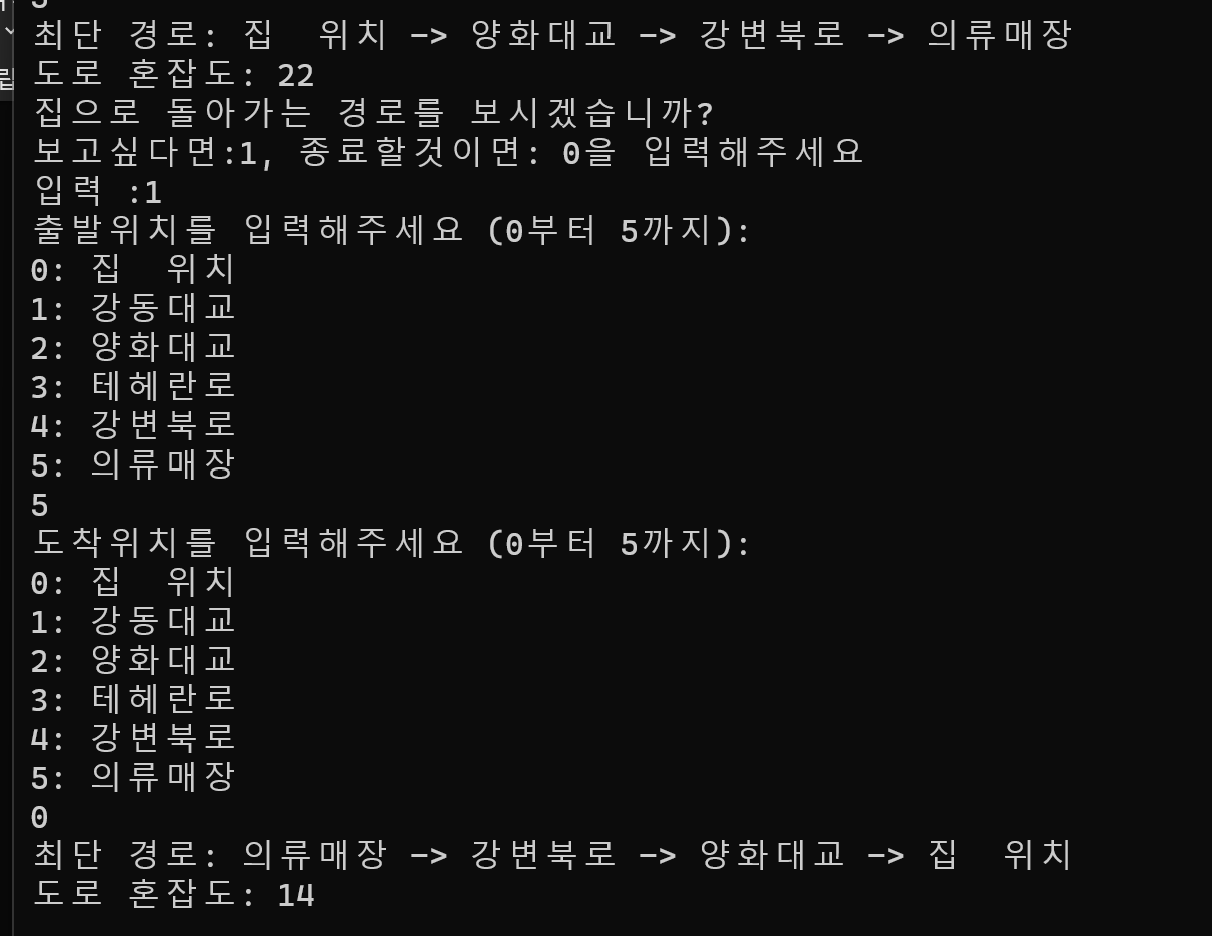
이후 계산이 종료되면,옷 매장까지 가는 최단경로는 Dijkstra알고리즘을 이용해주었습니다. 저는 vertex값을 index[0]에는 현재위치 index[5]에는 해당 옷 매장으로 설정해주었고, 나머지 index[1]~[4]는 도로의 이름을 명시해주었습니다. vertex값은 고정으로 두고, 교통상황에 따라 추천경로가 달라지는 것을 표현하고 싶어서 link에 할당된 weight의 값을 가변적으로 변할수있도록 rand()함수를 사용해줘서 사용자가 출력할 때마다 도로의 혼잡도가 바뀌어서 추천경로가 바뀌는 것을 구현해주고, 어떤 도로를 이용해야 최단 경로이고 도로의 혼잡도가 몇인지 출력되도록 프로그램을 구성해주었습니다. 그리고 사용자에게 추천경로 추천이후, 집으로 가는 경로를 추천 받을 것이냐고 물어본 뒤 아니라면 프로그램을 종료하고, 경로를 추천받는다면, dijstra알고리즘을 다시 사용해줘 집가는 경로를 출력하는 화면이 다시 출력되고 경로를 추천해주는 코드가 발생합니다



코드를 구성할 때, vertex의 개수를 6개로 정해주었고, link의 값도 9개로 고정시켜놨습니다.

Link당 가중되는 weight의 값은 도로교통상황, 혼잡도를 의미하며 그 값이 높을수록 도로교통도 혼잡하다는 것을 의미하며, 그 값이 낮을수록 도로가 복잡하지 않다는 것을 의미하여, 그 값이 낮은 길로 채택하여 최단거리를 구성할 수 있도록 하였습니다.

위 graph에서 무방향 그래프가 아닌 방향그래프로 설정해준 이유는, 같은 도로여도 어느방향으로 향하는지에 따라 혼잡도가 다르기 때문에 이를 표현해주고자 방향 그래프로 설정하기위해,인접행렬 [1][2]의 값과 인접행렬 [2][1]의 가중치가 서로 다르도록 설정해주었습니다. 이후 사용자에게 집 가는 경로를 물어보기 위해 첫번째 집위치에서 의류매장으로 가는 최단경로를 출력해준뒤

이처럼 입력창이 발생하게되고 1을 입력하면 의류매장에서 집위치까지의 최단경로를 추천해줍니다. 만약 0을 입력했다면 프로그램이 종료됩니다

2.코드설명

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#include <time.h>

#include <string.h>

#define MAX\_NODES 100

#define MAX\_LENGTH 100

typedef struct {

char name[50];

} Place;

typedef struct {

int vertex;//node

int weight2;//가중치

} Node;

typedef struct {

int numNodes;//vertex의 수

int numEdges;//link의 수

int adjacencyMatrix[MAX\_NODES][MAX\_NODES];//인접행렬은 정방행렬이다

Place places[MAX\_NODES];//장소의 배열

} Graph;

void printAdjacencyMatrix(Graph\* graph) {//인접행렬을 시각화하여 그리는 함수

printf("현재 교통상황 :\n");

printf(" ");

for (int i = 0; i < graph->numNodes; i++) {//graph의 노드수만큼 반복하여 이름을 출력한다

printf("%s ", graph->places[i].name);

}

printf("\n");

for (int i = 0; i < graph->numNodes; i++) {

printf("%s ", graph->places[i].name);//graph의 노드수만큼 반복하여 이름을 출력한다

for (int j = 0; j < graph->numNodes; j++) {

printf("%d ", graph->adjacencyMatrix[i][j]);//인접행렬에 있는 가중치를 출력한다

}

printf("\n");

}

}

int minDistance(int dist[], int visited[], int numNodes) {//dijkstra알고리즘에서사용 방문하지 않은 노드중 최소거리를 갖는 노드의 인덱스 반환

//,dist는 거리를,visited는 방문여부를 나타낸다

int min = INT\_MAX; //min은 최소거리를 저장하는 변수이다

int minIndex = -1; //minindex초기화,minIndex는 최소거리를 가지는 노드의인덱스를 저장

for (int i = 0; i < numNodes; i++) { //0부터 i<numNodes까지 반복하여 방문하지않은 노드중 최소거리를 갖는 노드를 찾는다

if (visited[i] == 0 && dist[i] <= min) {//만일 방문한적이 없고(==0), dist[i]의값이 현재의 min보다 작으면 dist[i]값을 min에할당

min = dist[i];

minIndex = i;//그 최솟값이 되는 i값을 minindex에 할당한다

}

}

return minIndex;

}

void printPath(Graph\* graph, int parent[], int j) {//dijkstra알고리즘에서 최단경로를 출력하는 함수,int parent는 부모의 배열을 나타낸다,rootnode

if (parent[j] == -1) //int parent가 -1이라면 rootnode인것이므로 함수를 종료한다

return;

printPath(graph, parent, parent[j]);//함수를 재귀적으로 출력하여 최단경로를 출력한다

printf("-> %s ", graph->places[j].name);//현재 노드의이름을 출력한다

}

void dijkstra(Graph\* graph, int startNode, int endNode) {

int dist[MAX\_NODES];//거리배열

int visited[MAX\_NODES];//방문한 노드배열

int parent[MAX\_NODES];//부모노드 배열

for (int i = 0; i < graph->numNodes; i++) {

dist[i] = INT\_MAX;//int\_max는 초기값으로 사용할때 모든 거리값을 초기화한다.dist는 최단거리

visited[i] = 0;//방문여부를 나타내는 배열.방문 안했으니 0으로 초기화

parent[i] = -1;//최상단 노드 즉 root node의 index는 -1로 표시한다,이후 최단경로 갱신될때마다 부모노드 갱신 union find다

}

dist[startNode] = 0;//시작노드의 최단거리를 0으로 초기화한다

for (int count = 0; count < graph->numNodes - 1; count++) {//count를 사용해서 각노드를 한번씩 모두 방문하는 반복문이다

int u = minDistance(dist, visited, graph->numNodes);//아직 방문하지 않은 노드중에서 최단거리 노드를 선택한다

visited[u] = 1;//방문한 노드는 1로 바꾼다

for (int v = 0; v < graph->numNodes; v++) {

if (visited[v] == 0 && graph->adjacencyMatrix[u][v] && dist[u] != INT\_MAX &&// 방문하지 않은 노드인지 확인하고

//graph가 해당 인접행렬에서 이어져있는지 확인

//최단거리가 무한대가 아닌지 확인한다

dist[u] + graph->adjacencyMatrix[u][v] < dist[v]) {//u를거쳐서 가는 최단거리+인접행렬의 가중치의합이 현재까지 최단거리보다 작다면

dist[v] = dist[u] + graph->adjacencyMatrix[u][v]; //최단거리를 u를 거쳐가도록 초기화시킨다

parent[v] = u;//v의 이전노드를 기록한다

}

}

}

printf("최단 경로: %s ", graph->places[startNode].name);//시작노드의 이름을 출력한다

printPath(graph, parent, endNode);//함수를 호출하여 시작노드부터 종료노드까지 최단경로를 출력,재귀적으로 이전노드를 출력한다

printf("\n도로 혼잡도: %d\n", dist[endNode]);//가중치의 최소합을 출력해준다

}

//스택구현함수

typedef struct {

char items[MAX\_LENGTH];

int top;

} Stack;//스택 구조체

void init(Stack\* stack) {//stack을 초기화시킨다

stack->top = -1;

}

int isEmpty(const Stack\* stack) {//stack이 공백이라면

return stack->top == -1;

}

int isFull(const Stack\* stack) {//stack이 가득찼다면

return stack->top == MAX\_LENGTH - 1;

}

void push(Stack\* stack, char item) {//stack에 새로운 값을 넣어준다

if (isFull(stack)) {

printf("스택이 가득 찼습니다.\n");

exit(EXIT\_FAILURE);

}

stack->items[++stack->top] = item;//stack의 초기는 -1이므로 1증가시키고 item을 넣어준다

}

char pop(Stack\* stack) {

if (isEmpty(stack)) {

printf("스택이 비어 있습니다.\n");

exit(EXIT\_FAILURE);

}

return stack->items[stack->top--];//top을 1감소시켜 stack에서 삭제시켜준다

}

char peek(const Stack\* stack) {

if (isEmpty(stack)) {

printf("스택이 비어 있습니다.\n");

exit(EXIT\_FAILURE);

}

return stack->items[stack->top];//stack에 최상단에 있는것을 확인한다

}

int operand(char ch) {

return (ch >= '0' && ch <= '9');//이함수는 문자CH가 0부터 9면 참을 반환,반환값이 0이면 숫자가 아닌다른문자

}

int precedence(char operator) {//문자를 받는 함수 각 연산자들을 반환한다

switch (operator) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

void infixToPostfix(const char\* infix, char\* postfix) {

Stack stack;//stack을 초기화하여주고

init(&stack);

int i, j;

for (i = 0, j = 0; infix[i] != '\0'; i++) {//중위 표기식을 한글자씩 읽는다

char token = infix[i];//i는 중위표기식의 문자열이다 token이라는 변수에 중위표기식 infix[i]를 저장하여

//문자열에서 읽어온 현재값을 나타냄

if (operand(token)) {//token이 피연산자인경우 operand함수가 true를 반환하는경우

postfix[j++] = token;//해당문자를 후위표기식 문자열인 postfix에 추가해주고 j를 증가시켜 다음위치를 가르킨다

}

else if (token == '(') {//문자가(라면 stack에 push하고

push(&stack, token);

}

else if (token == ')') {//문자가)라면

while (!isEmpty(&stack) && peek(&stack) != '(') {//스택이 비어잇지않고 스택의 최상단이 (가 아닐때까지 반복문실행

// '('입력되고')'출력될따까지 그사이 연산자들을 모두 pop시킨다

postfix[j++] = pop(&stack);

}

if (!isEmpty(&stack) && peek(&stack) != '(') {//'('들어오고 또 들어온게 '('라면 그 수식은 잘못된것이다

printf("잘못된 수식: 괄호 불일치\n");

exit(EXIT\_FAILURE);

}

pop(&stack); // '(' 제거

}

else {

while (!isEmpty(&stack) && precedence(token) <= precedence(peek(&stack))) {

postfix[j++] = pop(&stack);// stack이 공백이 아니고 스택의꼭대기에 있는 연산자의 우선순위가

//token의 우선순위보다 크거나 같을때,postfix를 추가하고 j를 증가시킨다

}

push(&stack, token);//이 pop된 stack들을 후위표기식에 push해준다

}

}

while (!isEmpty(&stack)) {//스택이 비어있지 않을때

postfix[j++] = pop(&stack);//stack을 pop하고 postfix에 추가한다 ,j값을 증가시킨뒤 해당위치에 문자를 추가

}

postfix[j] = '\0';//반복문이 종료되면 마지막 널문자 \0를 추가해 문자열 종료

}

int main() {

srand(time(NULL));

Graph graph;

graph.numNodes = 6;//vertex의 수

graph.numEdges = 9;//link의 수

for (int i = 0; i < graph.numNodes; i++) {//인접행렬의 값을 초기화

for (int j = 0; j < graph.numNodes; j++) {

graph.adjacencyMatrix[i][j] = 0;

}

}

strcpy(graph.places[0].name, "집 위치");

strcpy(graph.places[1].name, "강동대교");

strcpy(graph.places[2].name, "양화대교");

strcpy(graph.places[3].name, "테헤란로");

strcpy(graph.places[4].name, "강변북로");

strcpy(graph.places[5].name, "의류매장");

for (int i = 0; i < graph.numEdges; i++) {

int u = graph.numNodes;

int v = graph.numNodes;

int weight2 = rand() % 20;

graph.adjacencyMatrix[u][v] = weight2;

graph.adjacencyMatrix[v][u] = weight2;

}

double height, weight;

printf("신장(cm)을 입력하세요: ");

scanf("%lf", &height);

printf("체중(kg)을 입력하세요: ");

scanf("%lf", &weight);

printf("BMI의 계산식은 몸무게/키\*키입니다\n");

printf("컴퓨터가 연산을할때 후위표식으로 계산하게됩니다\n");

char infix[50];

printf("본인의 BMI계산식을 중위 표기식으로 입력하세요: ");

scanf("%s", infix);

char postfix[50];

infixToPostfix(infix, postfix);

printf("후위 표기식 : % s\n", postfix);

double heightMeter = height / 100.0;

double bmi = weight / (heightMeter \* heightMeter);

printf("BMI: %.2f\n", bmi);

if (bmi <= 18.5)

printf("저체중입니다\n");

else if (18.5 < bmi && bmi < 22.9)

printf("정상체중입니다\n");

else if (23 <= bmi && bmi < 24.9)

printf("과체중입니다\n");

else printf("비만입니다\n");

if (height <= 160) {

if (weight < 50)

printf("옷 사이즈 추천: XS\n");

else if (weight < 60)

printf("옷 사이즈 추천: S\n");

else if (weight < 70)

printf("옷 사이즈 추천: L\n");

else

printf("옷 사이즈 추천: XL\n");

}

else if (160 < height < 170) {

if (weight < 60)

printf("옷 사이즈 추천: M\n");

else if (weight < 70)

printf("옷 사이즈 추천: L\n");

else if (weight < 80)

printf("옷 사이즈 추천: XL\n");

else

printf("옷사이즈 추천:XXL");

}

else if (170 <= height < 180) {

if (weight < 60) {

printf("옷 사이즈 추천: M\n");

}

else if (60 < weight < 80)

printf("옷 사이즈 추천: XL\n");

else if (80 < weight < 100)

printf("옷 사이즈 추천: XXL\n");

else

printf("옷 사이즈 추천: XXL\n");

}

printf("해당 의류매장으로가는 최단경로를 계산해드리겠습니다\n");

for (int i = 0; i < graph.numNodes; i++) {

for (int j = 0; j < graph.numNodes; j++) {//link는 어디가 연결되고 어디가 연결안됐는지 고정이므로 해당 인접행렬을 직접선언해준다

if (i != j)

{

graph.adjacencyMatrix[0][1] = rand() % 20;

graph.adjacencyMatrix[0][2] = rand() % 20;

graph.adjacencyMatrix[1][0] = rand() % 20;

graph.adjacencyMatrix[1][2] = rand() % 20;

graph.adjacencyMatrix[1][3] = rand() % 20;

graph.adjacencyMatrix[1][4] = rand() % 20;

graph.adjacencyMatrix[2][0] = rand() % 20;

graph.adjacencyMatrix[2][1] = rand() % 20;

graph.adjacencyMatrix[2][3] = rand() % 20;

graph.adjacencyMatrix[2][4] = rand() % 20;

graph.adjacencyMatrix[3][1] = rand() % 20;

graph.adjacencyMatrix[3][2] = rand() % 20;

graph.adjacencyMatrix[3][5] = rand() % 20;

graph.adjacencyMatrix[4][1] = rand() % 20;

graph.adjacencyMatrix[4][2] = rand() % 20;

graph.adjacencyMatrix[4][5] = rand() % 20;

graph.adjacencyMatrix[5][3] = rand() % 20;

graph.adjacencyMatrix[5][4] = rand() % 20;

}// 11이상부터 20까지의 무작위 가중치

else

graph.adjacencyMatrix[i][j] = 0;

}//주대각행렬의 성분은 자기자신이므로 0이다

}

printAdjacencyMatrix(&graph);

int startNode, endNode;

printf("출발위치를 입력해주세요 (0부터 %d까지):\n", graph.numNodes - 1);

for (int i = 0; i < graph.numNodes; i++) {

printf("%d: %s\n", i, graph.places[i].name);

}

scanf("%d", &startNode);

printf("도착위치를 입력해주세요 (0부터 %d까지):\n", graph.numNodes - 1);

for (int i = 0; i < graph.numNodes; i++) {

printf("%d: %s\n", i, graph.places[i].name);

}

scanf("%d", &endNode);

dijkstra(&graph, startNode, endNode);

int restart=0;

printf("집으로 돌아가는 경로를 보시겠습니까? \n");

printf("보고싶다면:1, 종료할것이면: 0을 입력해주세요\n");

printf("입력 :");

scanf("%d",&restart);

if (restart == 1) {

int startNode, endNode;

printf("출발위치를 입력해주세요 (0부터 %d까지):\n", graph.numNodes - 1);

for (int i = 0; i < graph.numNodes; i++) {

printf("%d: %s\n", i, graph.places[i].name);

}

scanf("%d", &startNode);

printf("도착위치를 입력해주세요 (0부터 %d까지):\n", graph.numNodes - 1);

for (int i = 0; i < graph.numNodes; i++) {

printf("%d: %s\n", i, graph.places[i].name);

}

scanf("%d", &endNode);

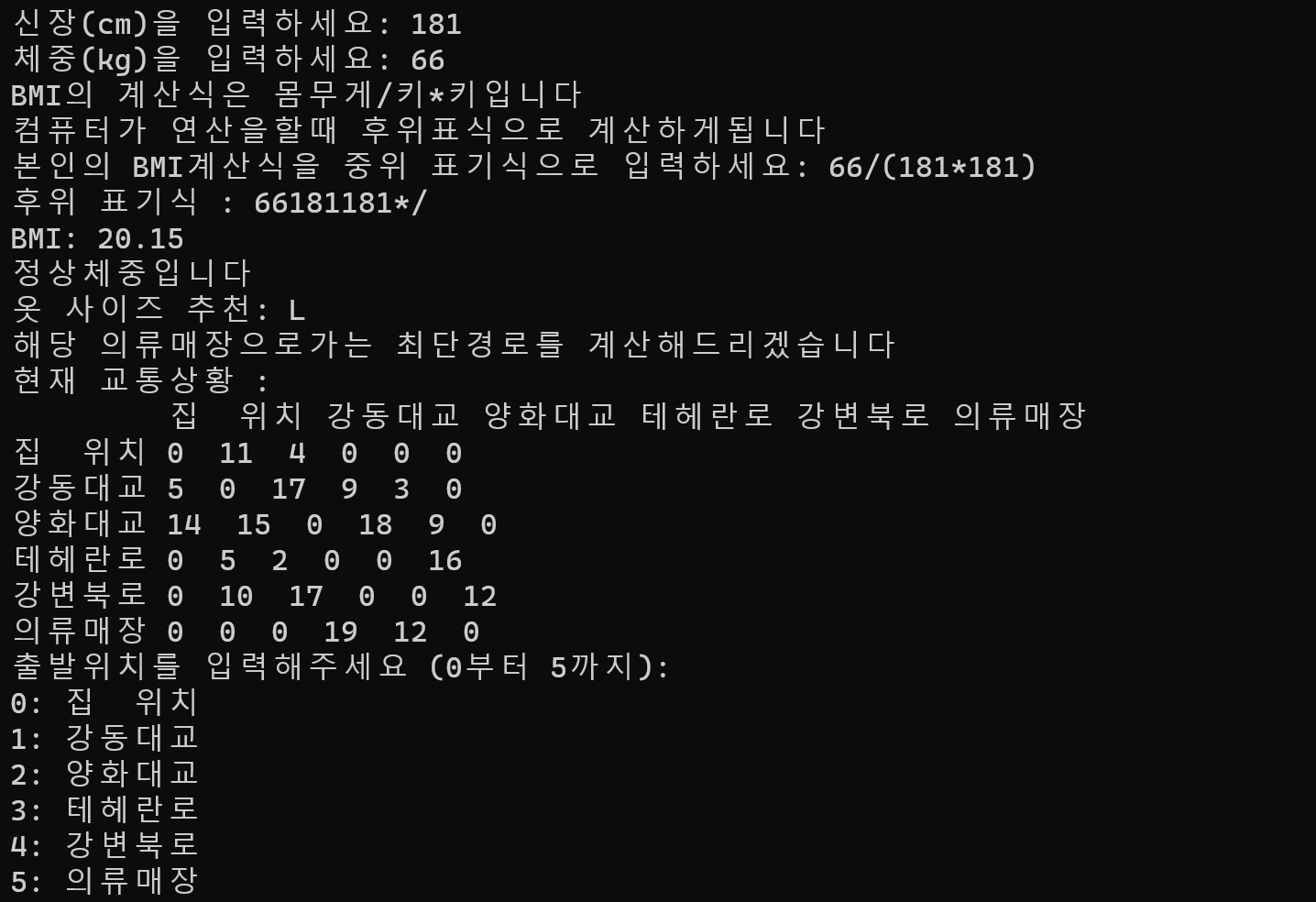
dijkstra(&graph, startNode, endNode);

}else

return 0;

}

3.출력화면



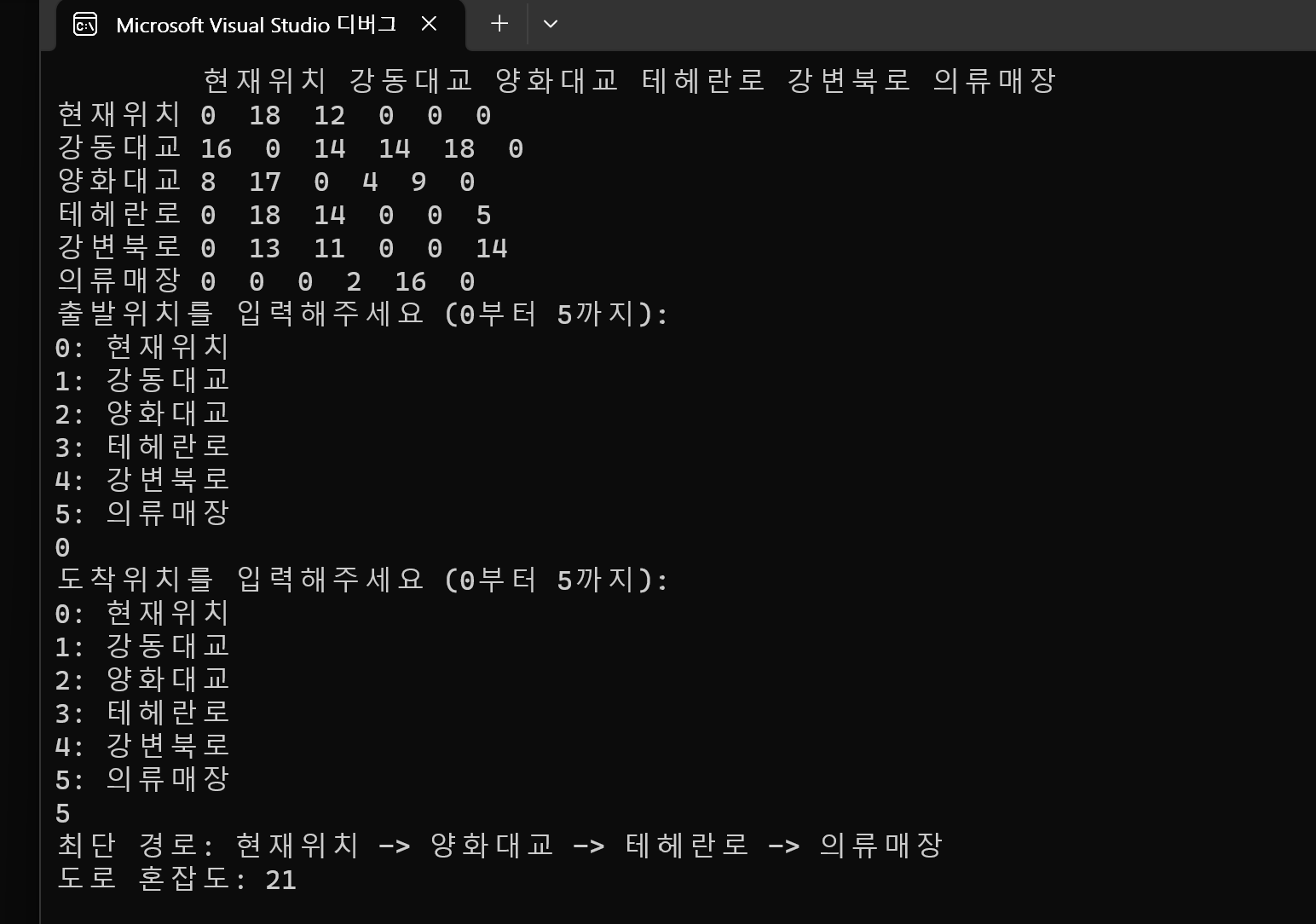
신장과 체중을 입력하고, 중위표기식으로 BMI계산식을 입력해주면 후위표기식이 stack으로 계산되게 됩니다.

이후 BMI값이 출력되고 BMI값에따라 정상체중, 저체중, 과체중이 판단되게되고

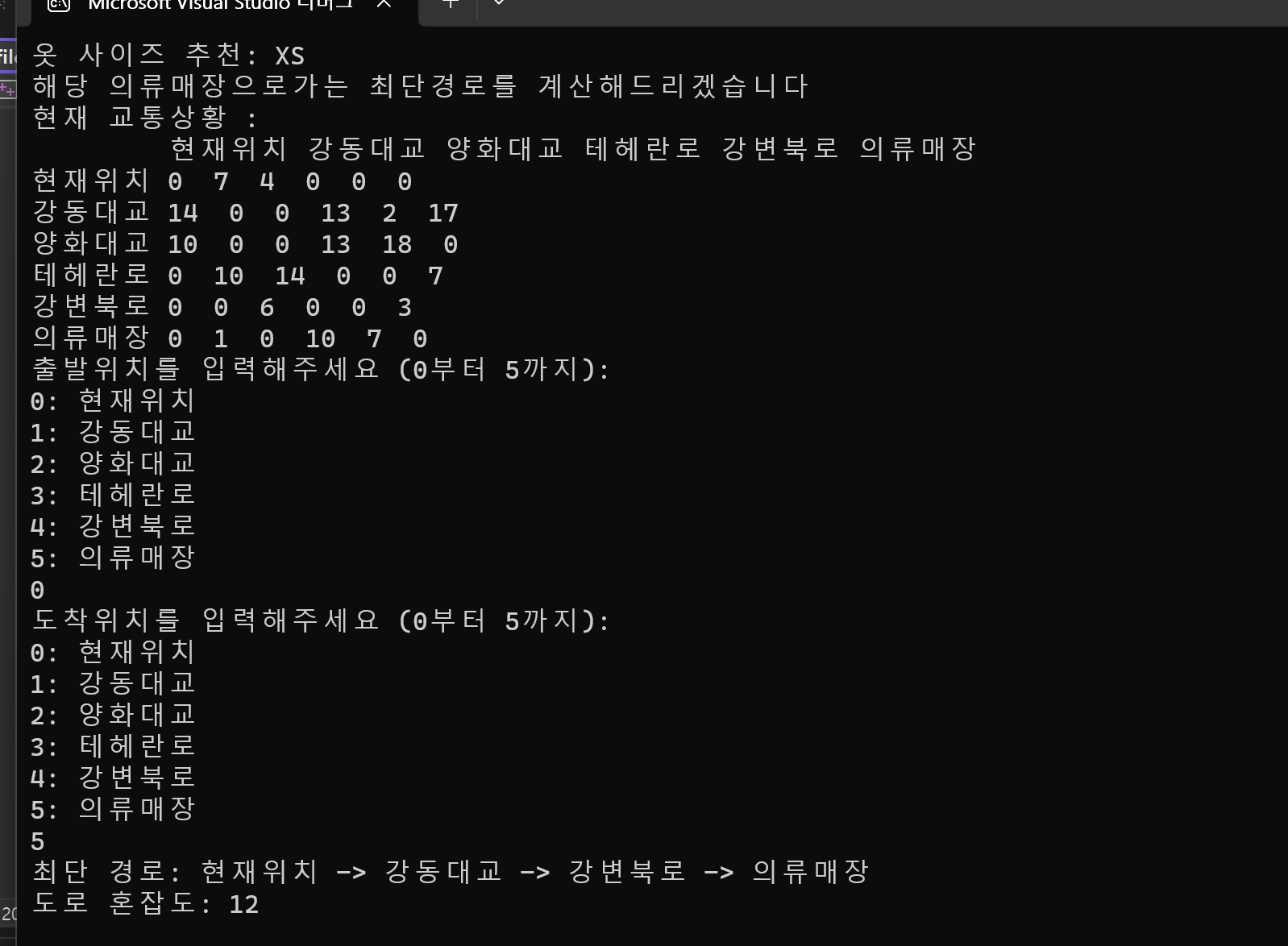
키 체중을 통하여 옷사이즈를 추천해주게 됩니다.

이후 의류매장으로가는 최단경로를 추천해주는데

인접행렬의 weight값이 난수로 발생되어 0부터 19까지 숫자가 생성되게 됩니다.



현재위치 0을 입력하고 가고자하는 의류매장 5를 입력하게되면 위와 같은 결과가 출력됩니다. 실행마다 Weight 값이 다르게 출력된다는 것을 보이기 위해, 아래 한번더 실행해보겠습니다.



난수를 발생시켜주었기 때문에 인접행렬의 값도 변화하고 최단경로를 다르게 추천해주고 도로혼잡도도 다르게 나타나는 것을 볼 수 있습니다.