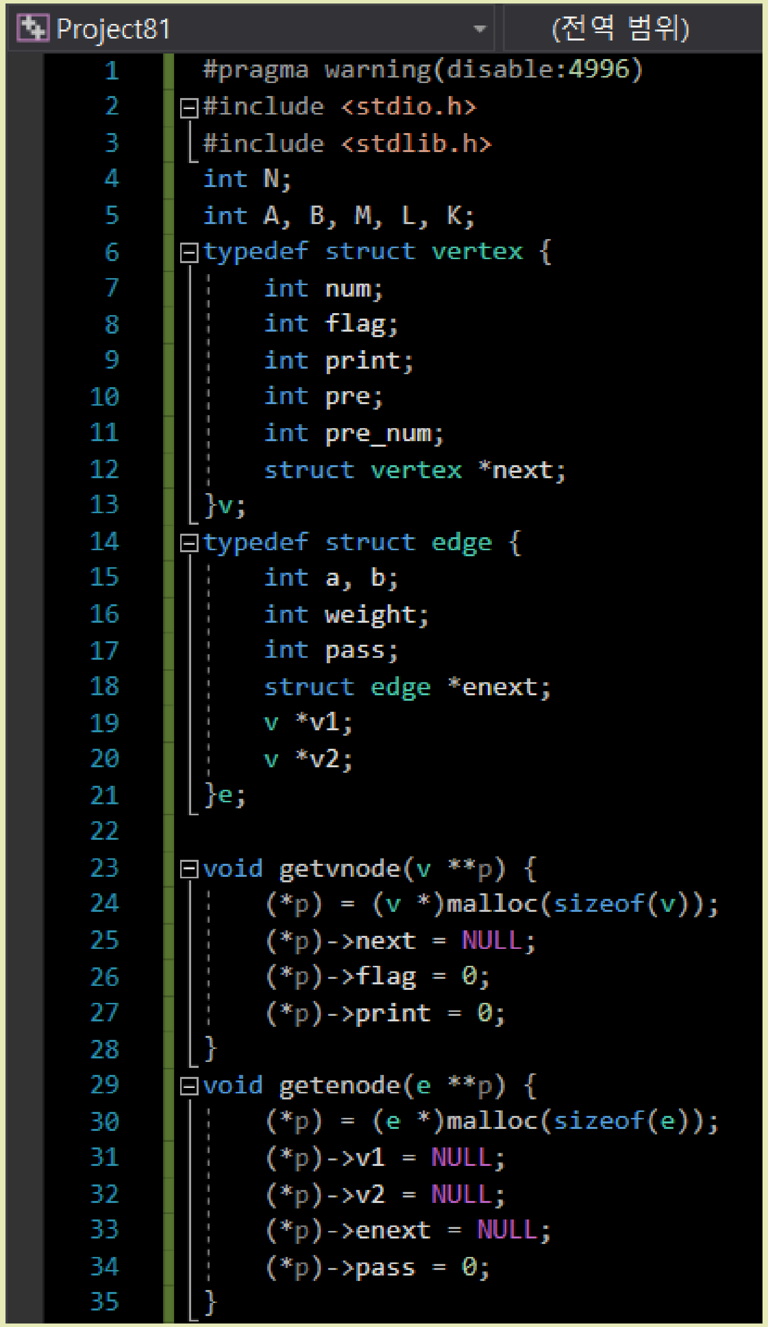
Term Project
Algorithm and Practices

19011824 이연희

Question 1
My code



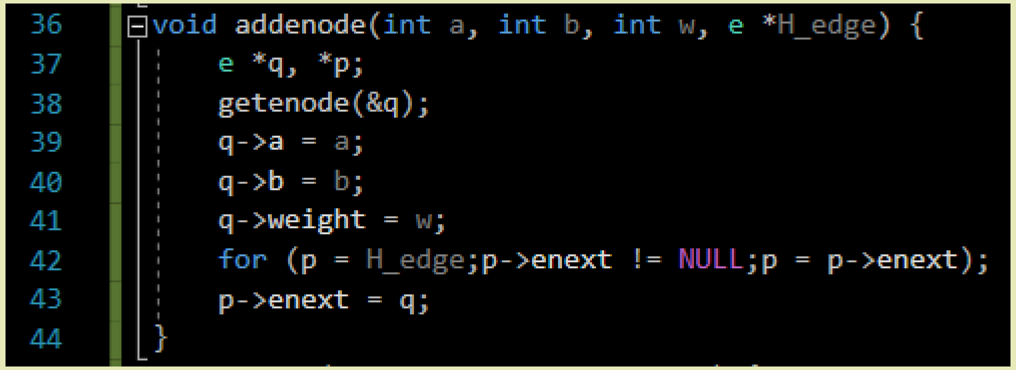
Project81 (전역 범위)

```c
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
int N;
int A, B, M, L, K;
typedef struct vertex {
	int num;
	int flag;
	int print;
	int pre;
	int pre_num;
	struct vertex *next;
}v;
typedef struct edge {
	int a, b;
	int weight;
	int pass;
	struct edge *enext;
	v *v1;
	v *v2;
}e;

void getvnode(v **p) {
	(*p) = (v *)malloc(sizeof(v));
	(*p)->next = NULL;
	(*p)->flag = 0;
	(*p)->print = 0;
}
void getenode(e **p) {
	(*p) = (e *)malloc(sizeof(e));
	(*p)->v1 = NULL;
	(*p)->v2 = NULL;
	(*p)->enext = NULL;
	(*p)->pass = 0;
}
```

I used list structures

num is “the number of a village or castle”
print is “how long it took from the starting point”
pre is “front num”
pre_num is “taking time from the preceding num”

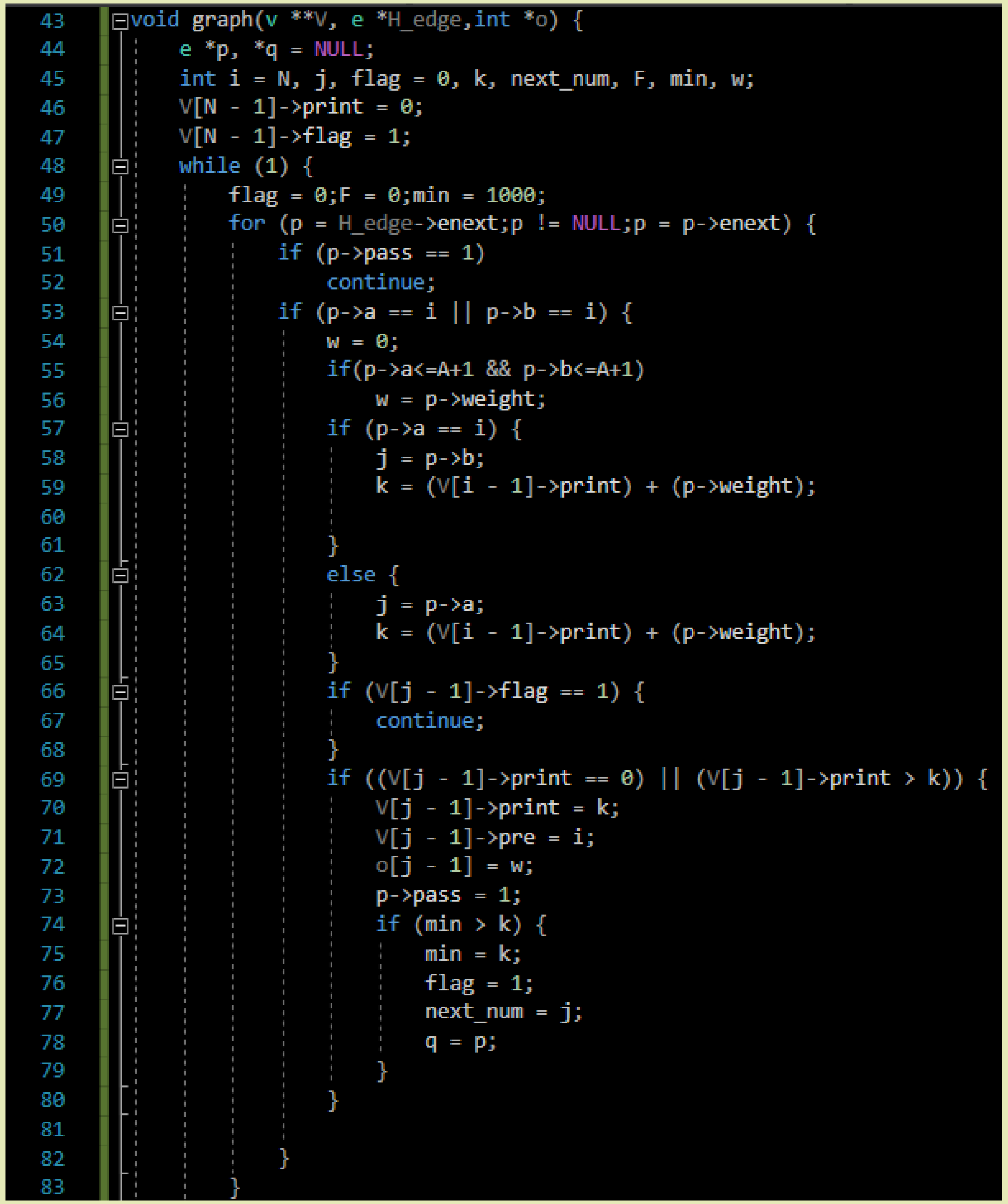
Code for adding path



```c
void addenode(int a, int b, int w, e *H_edge) {
	e *q, *p;
	getenode(&q);
	q->a = a;
	q->b = b;
	q->weight = w;
	for (p = H_edge;p->enext != NULL;p = p->enext);
	p->enext = q;
}
```

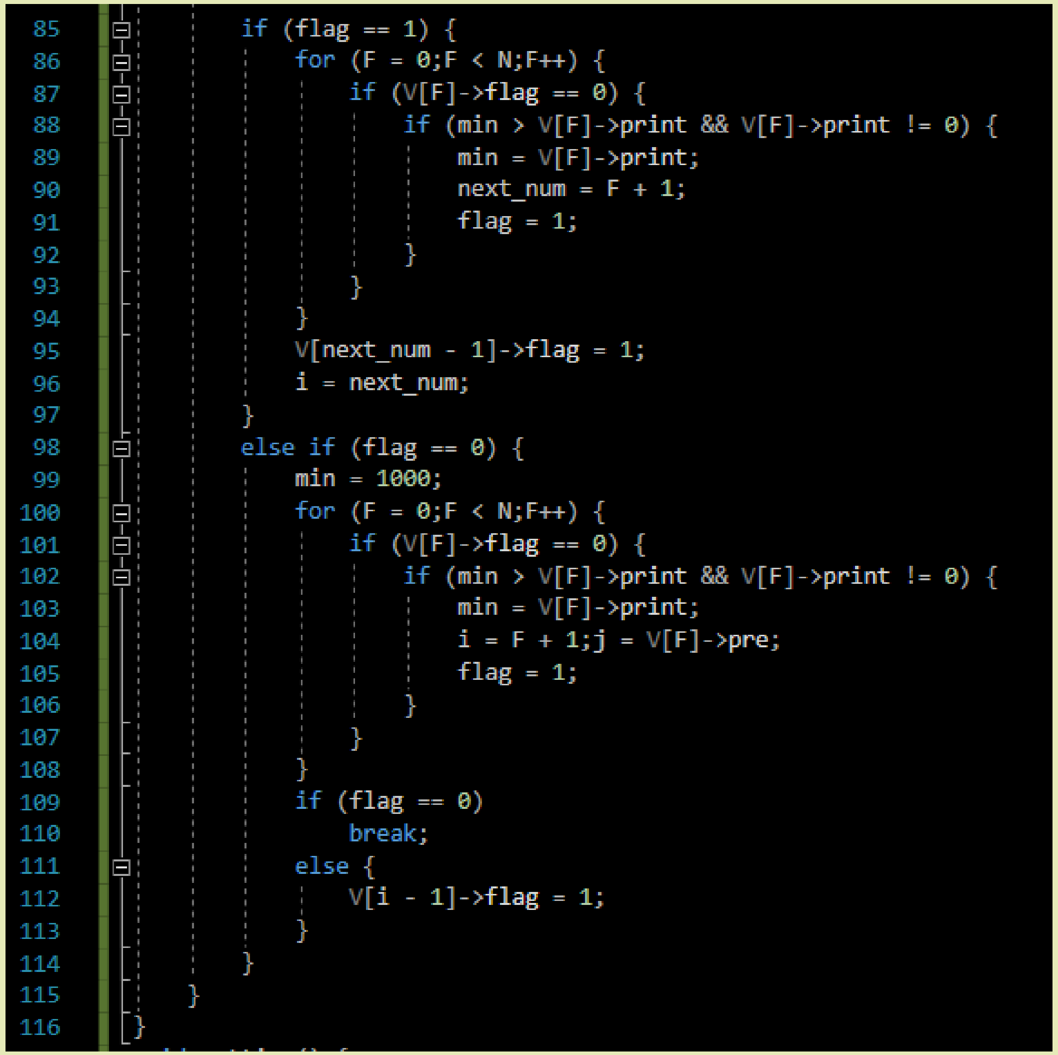
Question 1

My code



```c
void graph(v **V, e *H_edge,int *o) {
    e *p, *q = NULL;
    int i = N, j, flag = 0, k, next_num, F, min, w;
    V[N - 1]->print = 0;
    V[N - 1]->flag = 1;
    while (1) {
        flag = 0;F = 0;min = 1000;
        for (p = H_edge->enext;p != NULL;p = p->enext) {
            if (p->pass == 1)
                continue;
            if (p->a == i || p->b == i) {
                w = 0;
                if(p->a<=A+1 && p->b<=A+1)
                    w = p->weight;
                if (p->a == i) {
                    j = p->b;
                    k = (V[i - 1]->print) + (p->weight);

                }
                else {
                    j = p->a;
                    k = (V[i - 1]->print) + (p->weight);
                }
                if (V[j - 1]->flag == 1) {
                    continue;
                }
                if ((V[j - 1]->print == 0) || (V[j - 1]->print > k)) {
                    V[j - 1]->print = k;
                    V[j - 1]->pre = i;
                    o[j - 1] = w;
                    p->pass = 1;
                    if (min > k) {
                        min = k;
                        flag = 1;
                        next_num = j;
                        q = p;
                    }
                }

            }
        }
```

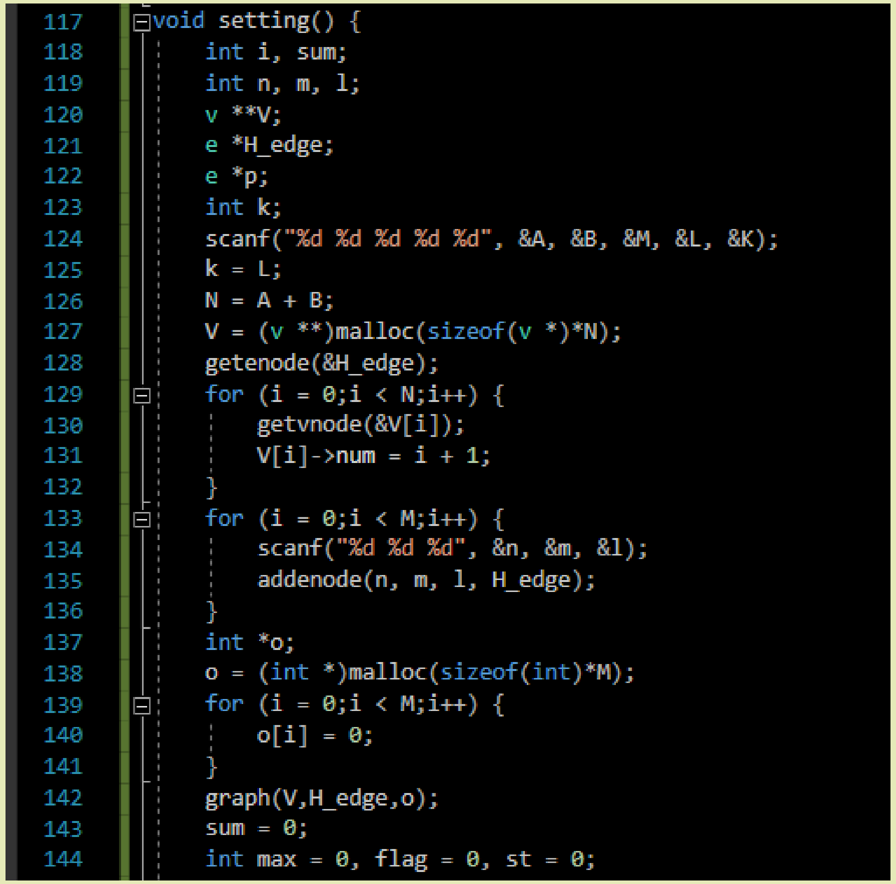
Code for finding minimum path



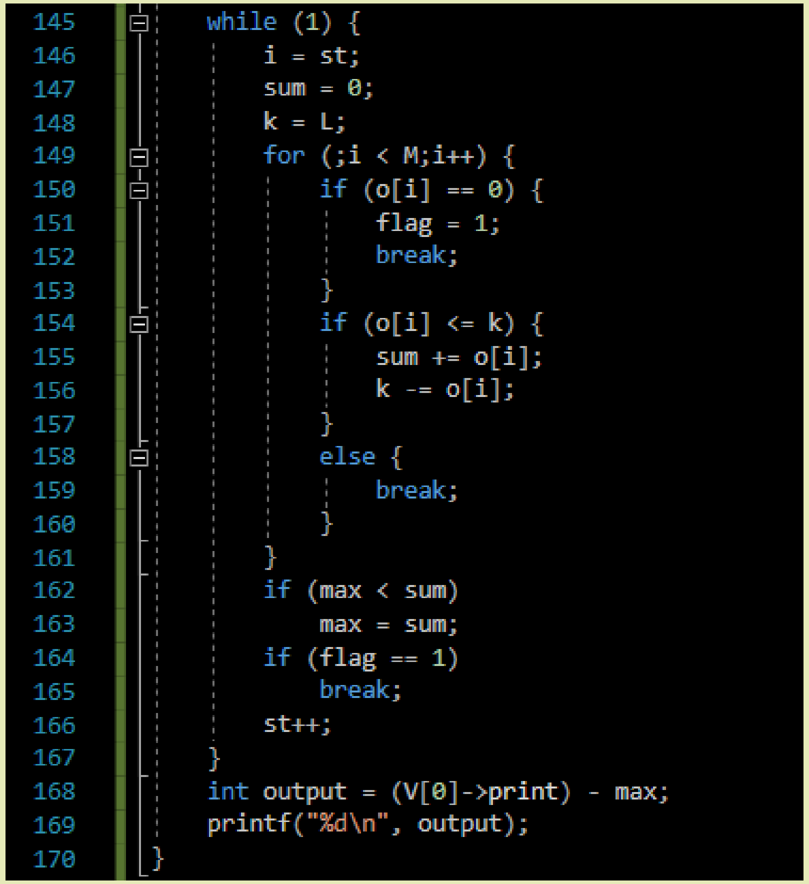
```c
        if (flag == 1) {
            for (F = 0;F < N;F++) {
                if (V[F]->flag == 0) {
                    if (min > V[F]->print && V[F]->print != 0) {
                        min = V[F]->print;
                        next_num = F + 1;
                        flag = 1;
                    }
                }
            }
            V[next_num - 1]->flag = 1;
            i = next_num;
        }
        else if (flag == 0) {
            min = 1000;
            for (F = 0;F < N;F++) {
                if (V[F]->flag == 0) {
                    if (min > V[F]->print && V[F]->print != 0) {
                        min = V[F]->print;
                        i = F + 1;j = V[F]->pre;
                        flag = 1;
                    }
                }
            }
            if (flag == 0)
                break;
            else {
                V[i - 1]->flag = 1;
            }
        }
    }
}
```

Question 1

My code

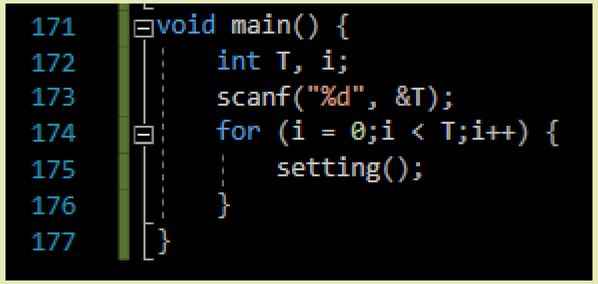


```
void setting() {
    int i, sum;
    int n, m, l;
    v **V;
    e *H_edge;
    e *p;
    int k;
    scanf("%d %d %d %d %d", &A, &B, &M, &L, &K);
    k = L;
    N = A + B;
    V = (v **)malloc(sizeof(v *)*N);
    getenode(&H_edge);
    for (i = 0;i < N;i++) {
        getvnode(&V[i]);
        V[i]->num = i + 1;
    }
    for (i = 0;i < M;i++) {
        scanf("%d %d %d", &n, &m, &l);
        addenode(n, m, l, H_edge);
    }
    int *o;
    o = (int *)malloc(sizeof(int)*M);
    for (i = 0;i < M;i++) {
        o[i] = 0;
    }
    graph(V,H_edge,o);
    sum = 0;
    int max = 0, flag = 0, st = 0;
```



```
    while (1) {
        i = st;
        sum = 0;
        k = L;
        for (;i < M;i++) {
            if (o[i] == 0) {
                flag = 1;
                break;
            }
            if (o[i] <= k) {
                sum += o[i];
                k -= o[i];
            }
            else {
                break;
            }
        }
        if (max < sum)
            max = sum;
        if (flag == 1)
            break;
        st++;
    }
    int output = (V[0]->print) - max;
    printf("%d\n", output);
}
```

Main code

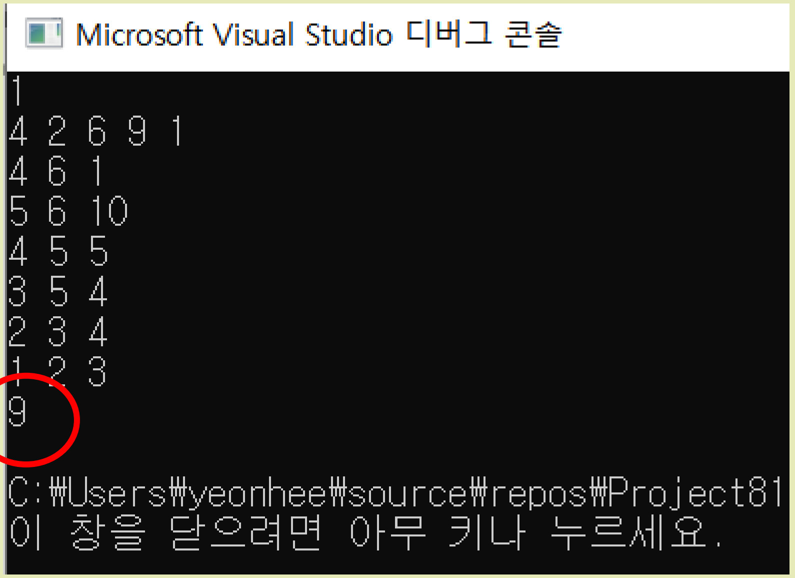


```
void main() {
    int T, i;
    scanf("%d", &T);
    for (i = 0;i < T;i++) {
        setting();
    }
}
```

Code to enter A,B,M,L,K and call a graph() function, and apply magic boots

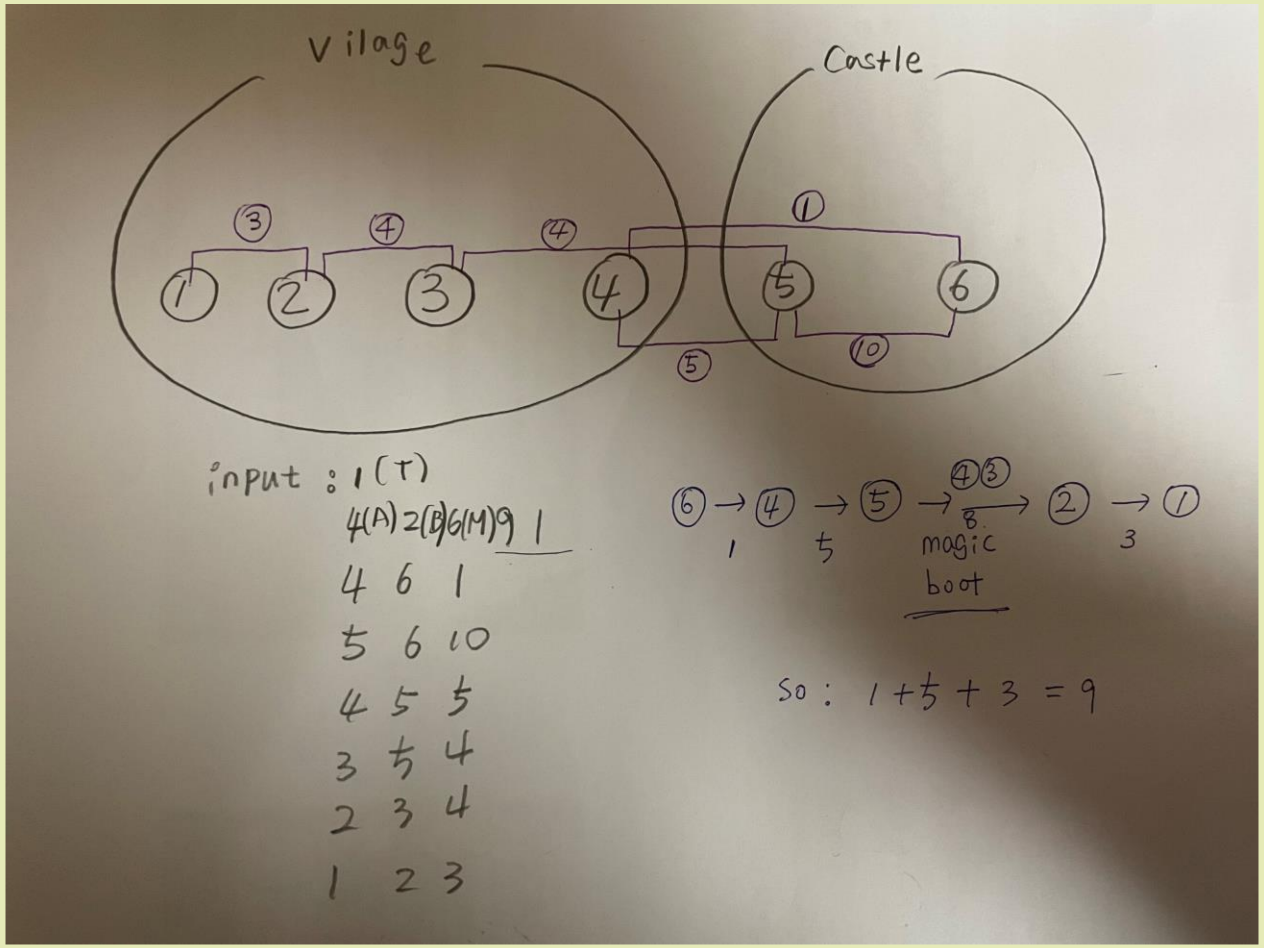
Question 1

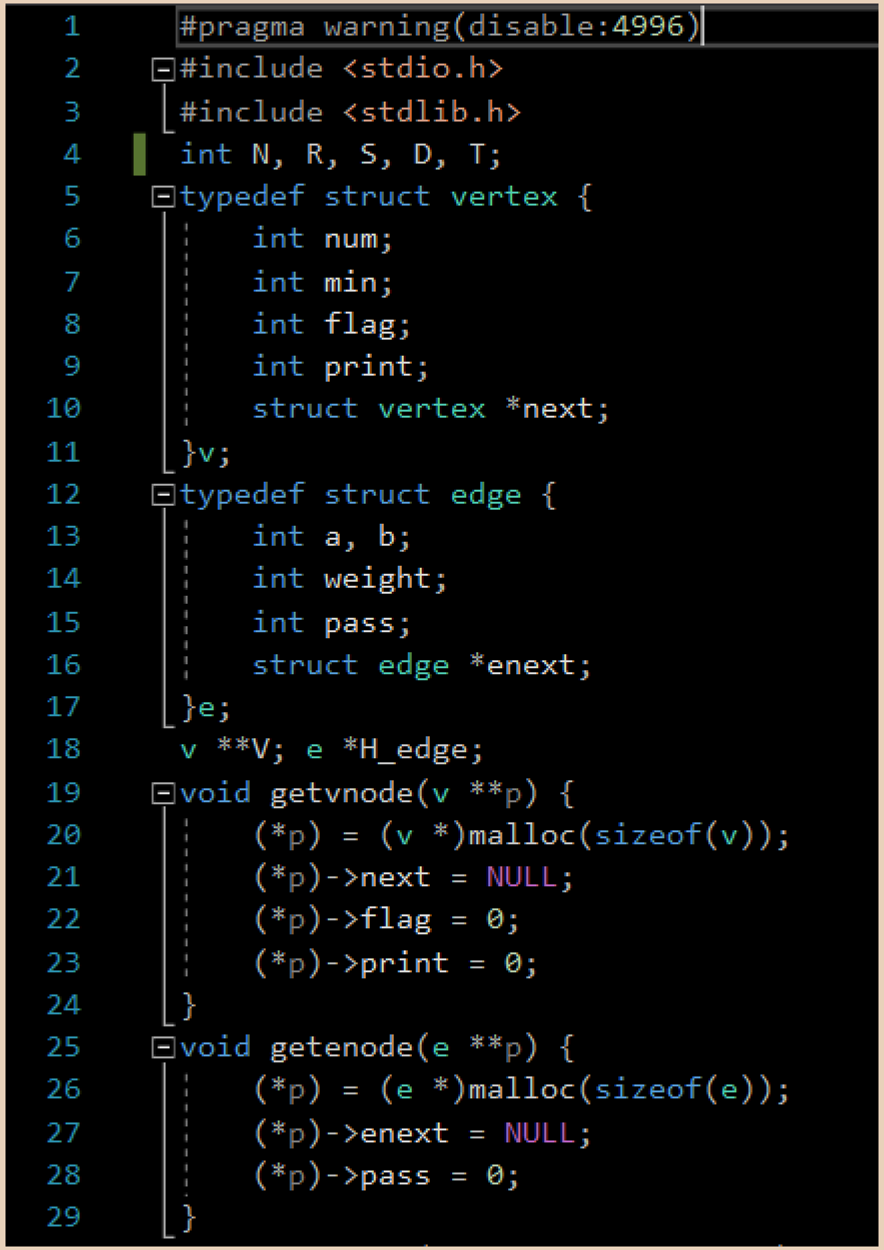
Result



Microsoft Visual Studio 디버그 콘솔

```
1
4 2 6 9 1
4 6 1
5 6 10
4 5 5
3 5 4
2 3 4
1 2 3
9

C:₩Users₩yeonhee₩source₩repos₩Project81
이 창을 닫으려면 아무 키나 누르세요.
```



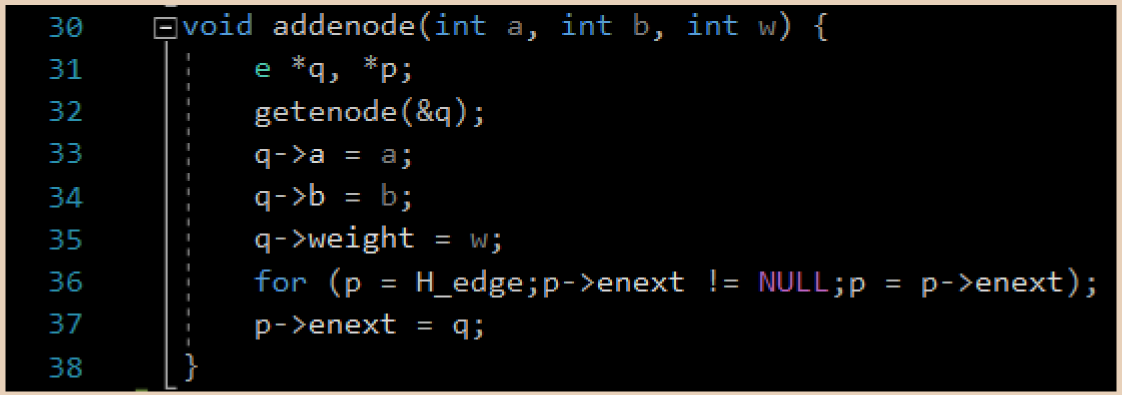


```c
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
int N, R, S, D, T;
typedef struct vertex {
    int num;
    int min;
    int flag;
    int print;
    struct vertex *next;
}v;
typedef struct edge {
    int a, b;
    int weight;
    int pass;
    struct edge *enext;
}e;
v **V; e *H_edge;
void getvnode(v **p) {
    (*p) = (v *)malloc(sizeof(v));
    (*p)->next = NULL;
    (*p)->flag = 0;
    (*p)->print = 0;
}
void getenode(e **p) {
    (*p) = (e *)malloc(sizeof(e));
    (*p)->enext = NULL;
    (*p)->pass = 0;
}
```

I used list structures

"min" has the smallest number of maximum capacity to reach the current peak.

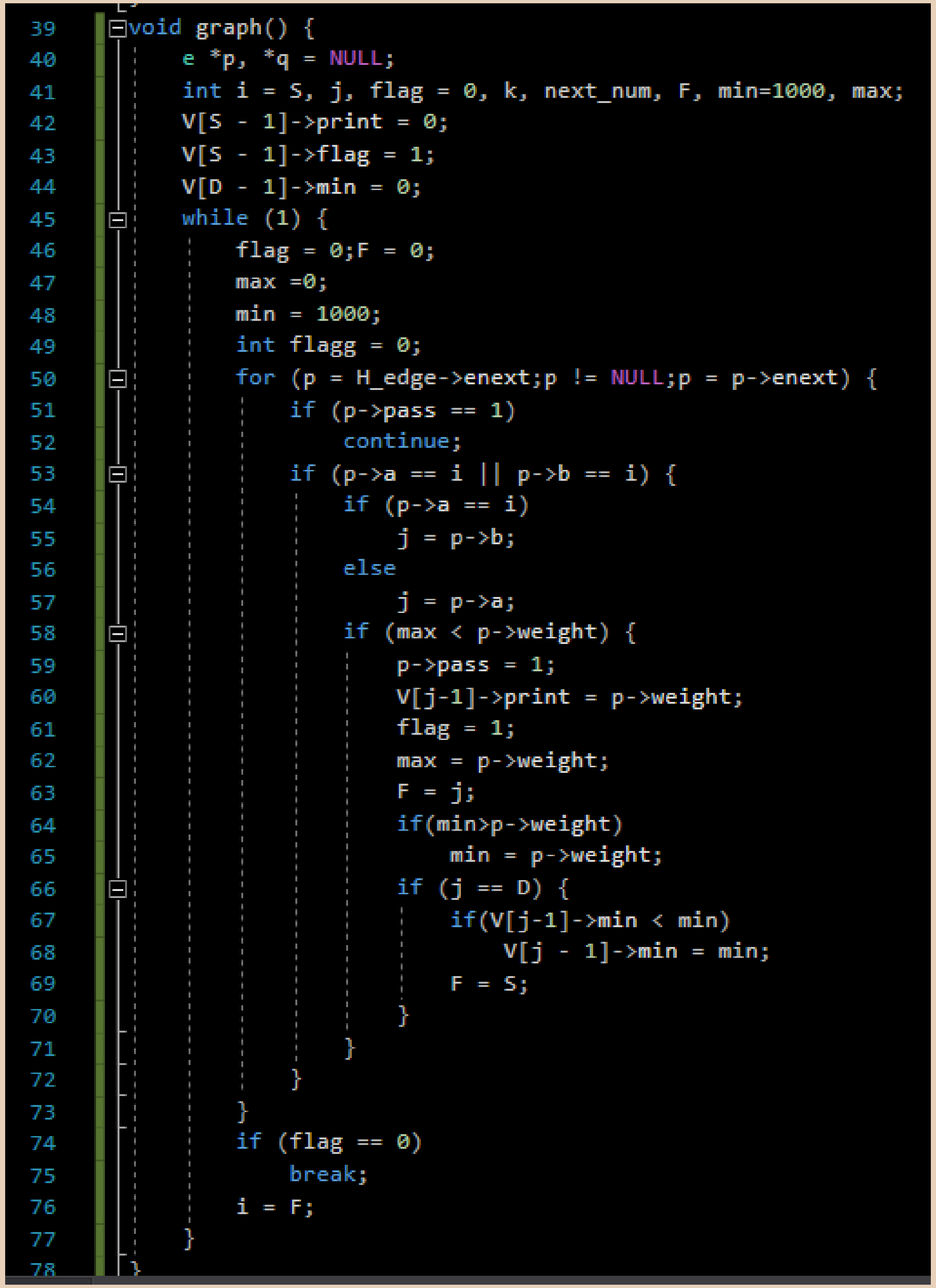
Code for adding path



```c
void addenode(int a, int b, int w) {
    e *q, *p;
    getenode(&q);
    q->a = a;
    q->b = b;
    q->weight = w;
    for (p = H_edge;p->enext != NULL;p = p->enext);
    p->enext = q;
}
```

Question 2

My code

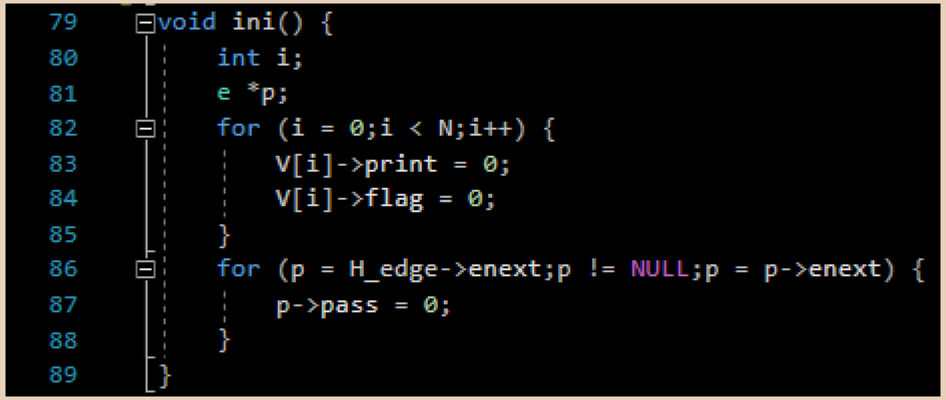


```
void graph() {
    e *p, *q = NULL;
    int i = S, j, flag = 0, k, next_num, F, min=1000, max;
    V[S - 1]->print = 0;
    V[S - 1]->flag = 1;
    V[D - 1]->min = 0;
    while (1) {
        flag = 0;F = 0;
        max =0;
        min = 1000;
        int flagg = 0;
        for (p = H_edge->enext;p != NULL;p = p->enext) {
            if (p->pass == 1)
                continue;
            if (p->a == i || p->b == i) {
                if (p->a == i)
                    j = p->b;
                else
                    j = p->a;
                if (max < p->weight) {
                    p->pass = 1;
                    V[j-1]->print = p->weight;
                    flag = 1;
                    max = p->weight;
                    F = j;
                    if(min>p->weight)
                        min = p->weight;
                    if (j == D) {
                        if(V[j-1]->min < min)
                            V[j - 1]->min = min;
                        F = S;
                    }
                }
            }
        }
        if (flag == 0)
            break;
        i = F;
    }
}
```

Code for finding the best route

Choose the route that can carry the most people from the vertex of S to the vertex of D.

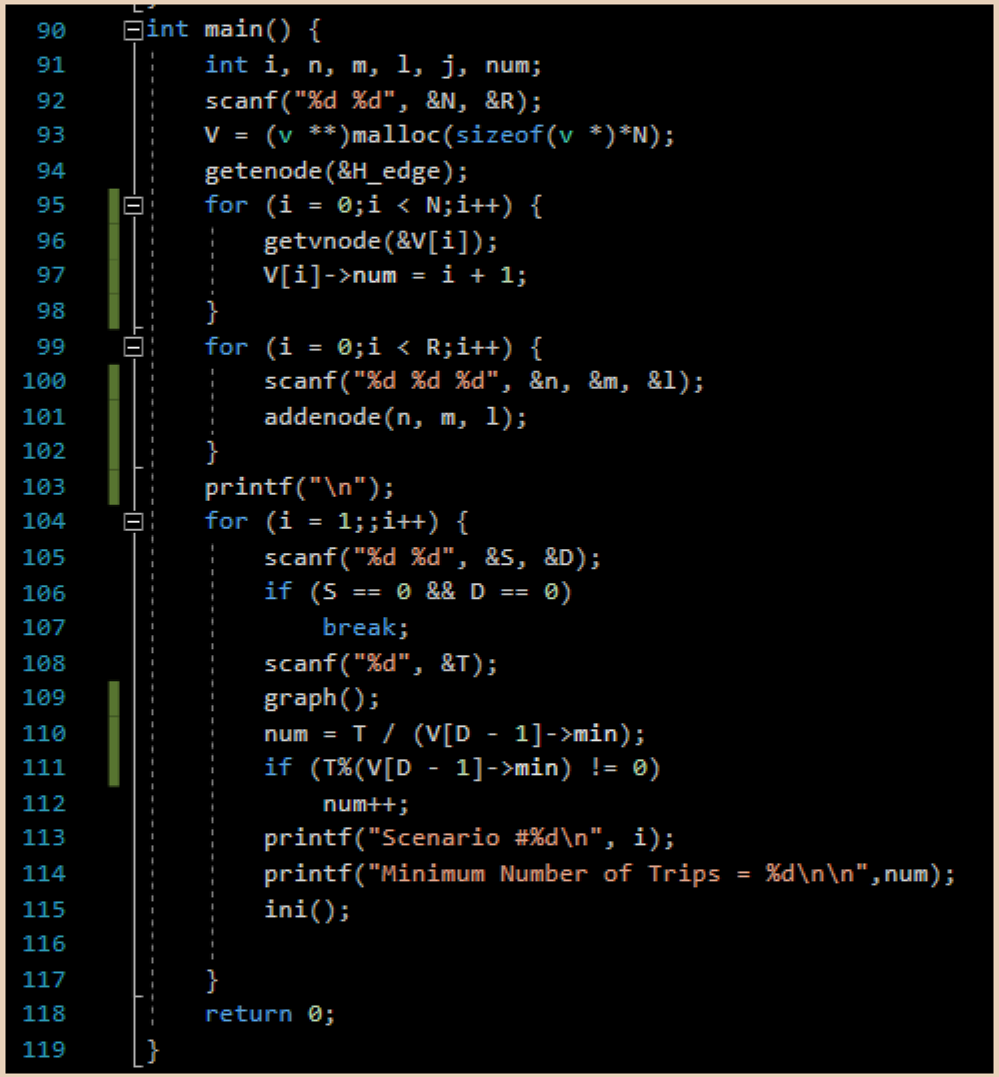
Initialization code for receiving the following input



```
void ini() {
    int i;
    e *p;
    for (i = 0;i < N;i++) {
        V[i]->print = 0;
        V[i]->flag = 0;
    }
    for (p = H_edge->enext;p != NULL;p = p->enext) {
        p->pass = 0;
    }
}
```

Question 2

My code



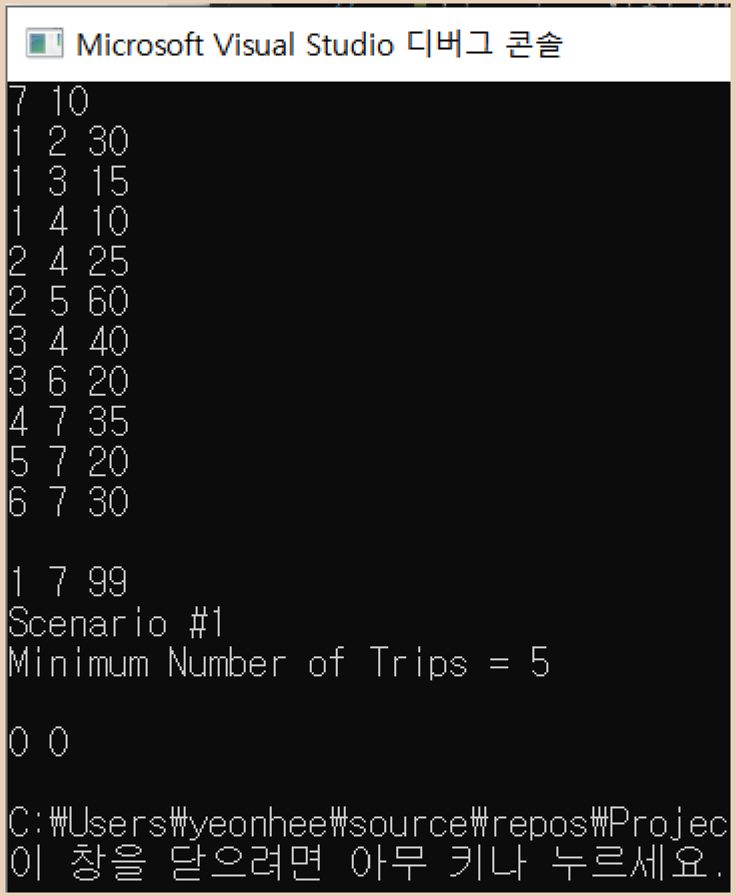
```c
int main() {
    int i, n, m, l, j, num;
    scanf("%d %d", &N, &R);
    V = (v **)malloc(sizeof(v *)*N);
    getenode(&H_edge);
    for (i = 0;i < N;i++) {
        getvnode(&V[i]);
        V[i]->num = i + 1;
    }
    for (i = 0;i < R;i++) {
        scanf("%d %d %d", &n, &m, &l);
        addenode(n, m, l);
    }
    printf("\n");
    for (i = 1;;i++) {
        scanf("%d %d", &S, &D);
        if (S == 0 && D == 0)
            break;
        scanf("%d", &T);
        graph();
        num = T / (V[D - 1]->min);
        if (T%(V[D - 1]->min) != 0)
            num++;
        printf("Scenario #%d\n", i);
        printf("Minimum Number of Trips = %d\n\n",num);
        ini();

    }
    return 0;
}
```

Main code

V is the vertex and N dynamic allocation.
Add path by calling up addnode() function.
Repeat until 0 is entered in S and D.
T is the number of people, and calls the graph() function to find the best path.
And get how many times guide go back and forth to num.

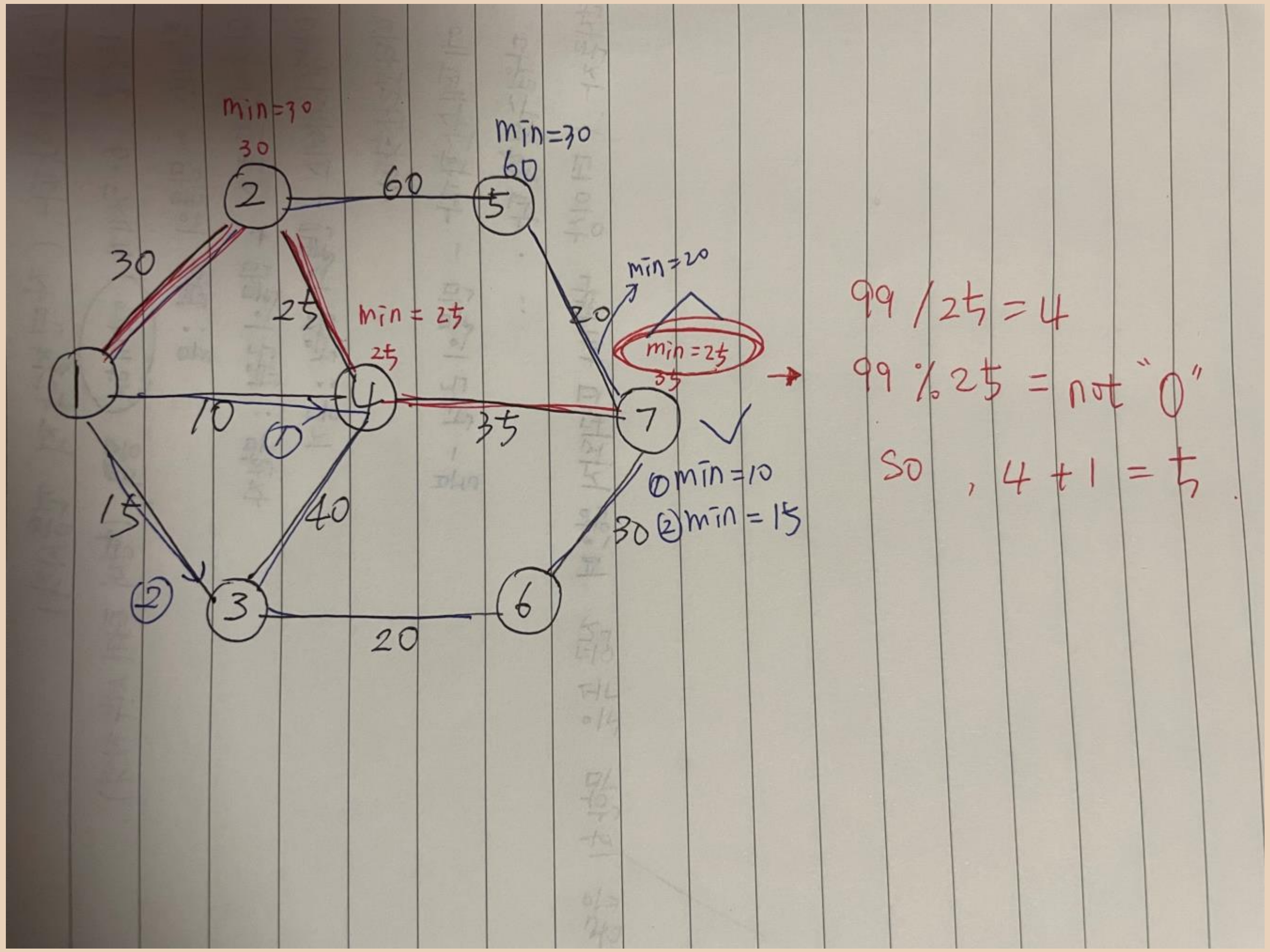
Question 2

Result



Microsoft Visual Studio 디버그 콘솔

```
7 10
1 2 30
1 3 15
1 4 10
2 4 25
2 5 60
3 4 40
3 6 20
4 7 35
5 7 20
6 7 30

1 7 99
Scenario #1
Minimum Number of Trips = 5

0 0

C:₩Users₩yeonhee₩source₩repos₩Projec
이 창을 닫으려면 아무 키나 누르세요.
```



Thank you ☺