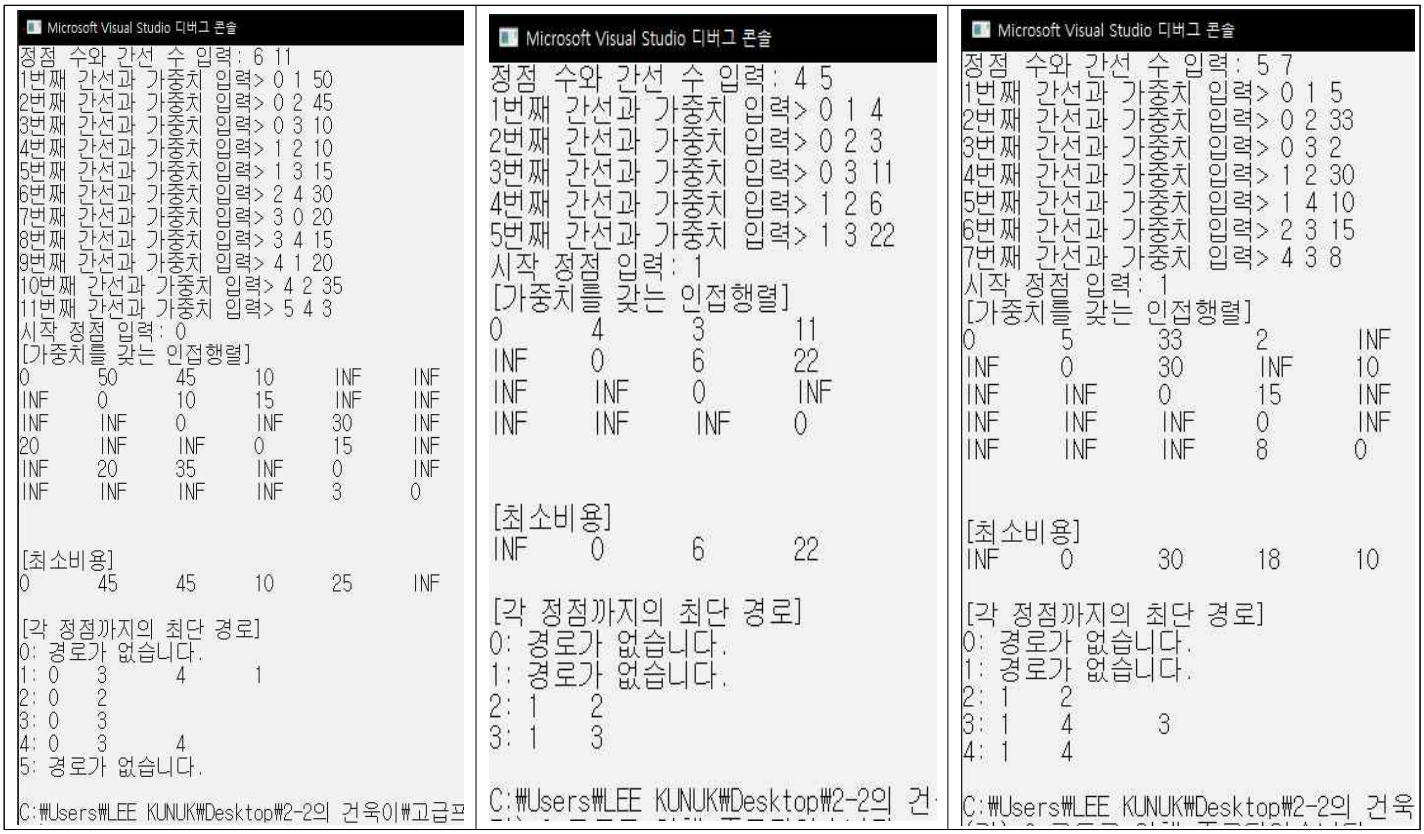
〈 소스코드 〉

```cpp
#include <iostream>
#include <algorithm>
#define infin 100000
using namespace std;

class Graph {
private:
    int **length;            // 가중치를 갖는 인접행렬
    int *dist;                 // 최단비용
    bool *s;                   //  방문했는지 기록
    int n;                     // 정점의 수
    int *path;
public:
    Graph(const int vertices = 0) : n(vertices)
    {
        length = new int*[n];   // 이차원 배열의 동적 생성
        for (int i = 0; i < n; i++)
        {
            length[i] = new int[n];
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (i == j) {
                    length[i][j] = 0;
                }
                else
                    length[i][j] = infin;
            }
        }
        path = new int[n];
        dist = new int[n];
        s = new bool[n];
        // 동적생성
    };
    void insertEdge(int, int, int);
    void ShortestPath(const int);
    int choose(const int);
    void Result(int start);
    void SearchPath(int, int);
};
void Graph::ShortestPath(const int v) {
    for (int i = 0; i < n; i++) {
        s[i] = false;
        dist[i] = length[v][i];
        path[i] = v;
    }
    s[v] = true;    // 시작지점
    dist[v] = 0;
    for (int i = 0; i < n - 2; i++) {
        int u = choose(n);  // 최솟값을 return
        s[u] = true;
        for (int w = 0; w < n; w++) {
            if (!s[w]) {
                if (dist[u] + length[u][w] < dist[w])
                {
                    dist[w] = dist[u] + length[u][w];
                    path[w] = u;    // 경로를 추가
                }
            }
        }
    }
}
int Graph::choose(const int v) {
    int min = infin;
    int index;
    for (int i = 0; i < v; i++) {
        if (s[i] != true)
        {
            if (dist[i] < min)
            {
                index = i;
                min = dist[i];
            }
        }
```

```
75          }
76          return index;   // dist가 최소가 되는 index값 리턴
77      };
78      void Graph::insertEdge(int p, int q, int w) {
79          length[p][q] = w;
80      };
81      void Graph::Result(int start) {     // 결과 출력하는 함수
82          cout << "[가중치를 갖는 인접행렬] " << endl;
83          for (int i = 0; i < n; i++) {
84              for (int j = 0; j < n; j++) {
85                  if (length[i][j] == infin)
86                      cout << "INF" << "\t";
87                  else
88                      cout << length[i][j] << "\t";
89              }
90              cout << endl;
91          }
92          cout << endl << endl;
93          cout << "[최소비용]" << endl;
94          for (int i = 0; i < n; i++) {
95              if (dist[i] == infin)
96                  cout << "INF" << "\t";
97              else
98                  cout << dist[i] << "\t";
99          }
100         cout << endl << endl;
101         cout << "[각 정점까지의 최단 경로]" << endl;
102         for (int i = 0; i < n; i++)
103         {
104             cout << i << ": ";
105             if (dist[i] == infin || i == start)
106                 cout << "경로가 없습니다. " << endl;
107             else
108             {
109                 SearchPath(i, start);
110                 cout << i << endl;
111             }
112         }
113     }
114     void Graph::SearchPath(int end, int start) {
115         if (path[end] != start)
116             SearchPath(path[end], start);
117         cout << path[end] << "\t";
118     }
119     int main(void) {
120         int n, m;
121         int x, y, weight, start;
122         cout << "정점 수와 간선 수 입력: ";
123         cin >> n >> m;
124
125         Graph G(n);
126
127         for (int i = 1; i <= m; i++) {
128             cout << i << "번째 간선과 가중치 입력> ";
129             cin >> x >> y >> weight;
130             G.insertEdge(x, y, weight);
131         }
132         cout << "시작 정점 입력: ";
133         cin >> start;
134         G.ShortestPath(start);
135         G.Result(start);
136
137         return 0;
138     }
139
```

〈 실행화면 〉



```
Microsoft Visual Studio 디버그 콘솔
정점 수와 간선 수 입력: 6 11
1번째 간선과 가중치 입력> 0 1 50
2번째 간선과 가중치 입력> 0 2 45
3번째 간선과 가중치 입력> 0 3 10
4번째 간선과 가중치 입력> 1 2 10
5번째 간선과 가중치 입력> 1 3 15
6번째 간선과 가중치 입력> 2 4 30
7번째 간선과 가중치 입력> 3 0 20
8번째 간선과 가중치 입력> 3 4 15
9번째 간선과 가중치 입력> 4 1 20
10번째 간선과 가중치 입력> 4 2 35
11번째 간선과 가중치 입력> 5 4 3
시작 정점 입력: 0
[가중치를 갖는 인접행렬]
0       50      45      10      INF     INF
INF     0       10      15      INF     INF
INF     INF     0       INF     30      INF
20      INF     INF     0       15      INF
INF     20      35      INF     0       INF
INF     INF     INF     INF     3       0

[최소비용]
0       45      45      10      25      INF

[각 정점까지의 최단 경로]
0: 경로가 없습니다.
1: 0    3       4       1
2: 0    2
3: 0    3
4: 0    3       4
5: 경로가 없습니다.

C:₩Users₩LEE KUNUK₩Desktop₩2-2의 건욱이₩고급프
```

```
Microsoft Visual Studio 디버그 콘솔
정점 수와 간선 수 입력: 4 5
1번째 간선과 가중치 입력> 0 1 4
2번째 간선과 가중치 입력> 0 2 3
3번째 간선과 가중치 입력> 0 3 11
4번째 간선과 가중치 입력> 1 2 6
5번째 간선과 가중치 입력> 1 3 22
시작 정점 입력: 1
[가중치를 갖는 인접행렬]
0       4       3       11
INF     0       6       22
INF     INF     0       INF
INF     INF     INF     0


[최소비용]
INF     0       6       22

[각 정점까지의 최단 경로]
0: 경로가 없습니다.
1: 경로가 없습니다.
2: 1    2
3: 1    3

C:₩Users₩LEE KUNUK₩Desktop₩2-2의 건
```

```
Microsoft Visual Studio 디버그 콘솔
정점 수와 간선 수 입력: 5 7
1번째 간선과 가중치 입력> 0 1 5
2번째 간선과 가중치 입력> 0 2 33
3번째 간선과 가중치 입력> 0 3 2
4번째 간선과 가중치 입력> 1 2 30
5번째 간선과 가중치 입력> 1 4 10
6번째 간선과 가중치 입력> 2 3 15
7번째 간선과 가중치 입력> 4 3 8
시작 정점 입력: 1
[가중치를 갖는 인접행렬]
0       5       33      2       INF
INF     0       30      INF     10
INF     INF     0       15      INF
INF     INF     INF     0       INF
INF     INF     INF     8       0


[최소비용]
INF     0       30      18      10

[각 정점까지의 최단 경로]
0: 경로가 없습니다.
1: 경로가 없습니다.
2: 1    2
3: 1    4       3
4: 1    4

C:₩Users₩LEE KUNUK₩Desktop₩2-2의 건욱
```