Given an undirected weighted graph find the shortest path of node 1 from the node 0, after subtracting a particular value (offset) from the weight of every edge.

For this problem, someone has submitted the following code. The idea of the program is given briefly:
1. First run the dijkstra from 0 to 1. This will give the shortest path length for the input graph.
2. Then run the bfs from 0 to 1. This will give the number of edges on the path of 0 to 1.
3. Each of the edge weights on step 3 will be reduced by the amount of offset.
4. The final result is the value from step 1 minus the value from step2 multiplied by offset.

```
#include<bits/stdc++.h>
using namespace std;
#define inf (1<<29)
#define ii64 long long
#define MX 100002
struct node{
    int nd, cost;
    node(){}
    node(int _nd,int _cost){
        nd = _nd;
        cost = _cost;
    }
    bool operator >(const node &a)const{
        return cost > a.cost;
    }
};
vector<node> g[MX];
int d[MX];
int dijkstra(int n,int source,int dest){
               priority_queue<node,    vector<node>,
greater<node> > q;
    for(int i = 0; i < n; i++){
        d[i] = inf;
    }
    while(!q.empty())q.pop();
    d[source] = 0;
    q.push(node(source,0));
    while(!q.empty()){
        int u = q.top().nd;
        q.pop();
        for(int i = 0; i < g[u].size(); i++){
            int v = g[u][i].nd;
            int w = g[u][i].cost;
            if(d[v] > d[u] + w){
                d[v] = d[u] + w;
                q.push(node(v,d[v]));
            }
        }
    }
    return d[dest];
}
```

```
int bfs(int n,int source,int dest){
    queue<node> q;
    for(int i = 0; i < n; i++){
        d[i] = inf;
    }
    while(!q.empty())q.pop();
    d[source] = 0;
    q.push(node(source,0));
    while(!q.empty()){
        int u = q.front().nd;
        q.pop();
        for(int i = 0; i < g[u].size(); i++){
            int v = g[u][i].nd;
            if(d[v] > d[u] + 1){
                d[v] = d[u] + 1;
                q.push(node(v,d[v]));
            }
        }
    }
    return d[dest];
}
int main(){
    int n,m,u,v,w,offset,res;
    scanf("%d %d",&n, &m);
    while(m--){
        scanf("%d %d %d",&u,&v,&w);
        g[u].push_back(node(v,w));
        g[v].push_back(node(u,w));
    }
    scanf("%d",&offset);
    res = dijkstra(n,0,1) - bfs(n,0,1)*offset;
    printf("%d\n",res);
    return 0;
}
```

But this code is actually wrong for some test case. Find one such case.

Input
There is no input.

Output
Write one program that will print the case in the given format. First line: Number of nodes, N. Second line: Number of edges. Next M lines each U V W (an edge between U and V with edge W). Next line: Offset. You need to maintain the following constraints:
1. 10 <= N <=15
2. 10 <= M <=25
3. 0 <= U,V < N
4. 0 < W <=100
5. Value of the offset must be such that all the edge weights stay positive even after the subtraction.
6. The graph must be connected.

The sample output is not correct. Its just here to show you the format.

No input	7 6 4 3 2 4 2 5 0 4 3 3 2 2 3 6 50 2 1 10 1