```
/*********   dijkstra special by Shadman ******/
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
#define ll long long
using namespace std;
const int MAXN=110;
const int MAXE=2010;
const int INF=999999999;
int n,m,l;
struct EDGE
{
    int v,next;
    int dis;
}edge[MAXE];
struct HeadNode
{
    int d,u;
    bool operator < (const HeadNode& rhs) const
    {
        return d>rhs.d;
    }
};
int head[MAXN],size,pre[MAXN];
ll sum[MAXN],ans0,ans,tmp;
bool bel[MAXN][MAXE],del[MAXE];
void init()
{
    memset(head,-1,sizeof(head));
    size=0;
}
void add_edge(int u,int v,int dis)
{
    edge[size].v=v;
    edge[size].dis=dis;
    edge[size].next=head[u];
    head[u]=size++;
}
int dist[MAXN];
bool vis[MAXN];
void dijkstra(int s)
{
    memset(pre,-1,sizeof(pre));
    memset(vis,0,sizeof(vis));
    priority_queue<HeadNode> q;
    for(int i=1;i<=n;i++)
        dist[i]=INF;
    dist[s]=0;
    sum[s]=0;
    q.push(HeadNode{0,s});
    while(!q.empty())
    {
        HeadNode x=q.top();
        q.pop();
        int u=x.u;
        if(vis[u])
            continue;
```

```
        vis[u]=1;
        for(int i=head[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].v;
            if(dist[v]>dist[u]+edge[i].dis)
            {
                dist[v]=dist[u]+edge[i].dis;
                pre[v]=i;
                q.push(HeadNode{dist[v],v});
            }
        }
    }
    for(int i=1;i<=n;i++)
    {
        if(i!=s)
        {
            if(dist[i]<INF)
                bel[s][pre[i]]=bel[s][pre[i]^1]=1;
            else
                dist[i]=l;
            ans0+=dist[i];
            sum[s]+=dist[i];
        }
    }
}
void dijkstra1(int s)
{
    memset(vis,0,sizeof(vis));
    priority_queue<HeadNode> q;
    for(int i=1;i<=n;i++)
        dist[i]=INF;
    dist[s]=0;
    q.push(HeadNode{0,s});
    while(!q.empty())
    {
        HeadNode x=q.top();
        q.pop();
        int u=x.u;
        if(vis[u])
            continue;
        vis[u]=1;
        for(int i=head[u];i!=-1;i=edge[i].next)
        {
            if(del[i])
                continue;
            int v=edge[i].v;
            if(dist[v]>dist[u]+edge[i].dis)
            {
                dist[v]=dist[u]+edge[i].dis;
                q.push(HeadNode{dist[v],v});
            }
        }
    }
    ll k=0;
    for(int i=1;i<=n;i++)
    {
        if(i!=s)
        {
```

```
            if(dist[i]==INF)
                dist[i]=l;
            k+=dist[i];
        }
    }
    tmp=tmp-sum[s]+k;
}
int main()
{
    int i;
    while(scanf("%d%d%d",&n,&m,&l)!=EOF)
    {
        int u,v,d;
        init();
        while(m--)
        {
            scanf("%d%d%d",&u,&v,&d);
            add_edge(u,v,d);
            add_edge(v,u,d);
        }
        ans0=0;
        memset(bel,0,sizeof(bel));
        for(i=1;i<=n;i++)
            dijkstra(i);
        printf("%lld ",ans0);
        ans=ans0;
        memset(del,0,sizeof(del));
        for(int i=0;i<size;i+=2)
        {
            del[i]=del[i^1]=1;
            tmp=ans0;
            for(int j=1;j<=n;j++)
                if(bel[j][i])
                dijkstra1(j);
            del[i]=del[i^1]=0;
            ans=max(ans,tmp);
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```