魔女之旅

题意：

一共有$N+1$个城市（如果把伊蕾娜家也算上的话），每个城市的纸价不尽相同，和$N+1$条路，经过每条路的消耗纸张量也不尽相同，求最小花费。

分析：

路程len_i和单位路程消耗的纸张m_i之积实际上就是这条路所消耗的纸张总量tot_i，如果$tot_i > K$，那么我们的伊蕾娜就无法到达终点，此时我们程序就该输出#-1

如果**每张纸的价格最低**，那么我们的总价也会最低，这样就证明了贪心的正确性。

假设现在我们到了第i个城市，接下来我们有三种情况可以进行分类讨论：

①在我们能够到达的**最远处**（即我们买满纸所能到达的地方）之内，有比**当前城市**纸价更低的**第一个**地方，我们就在**当前城市**买刚好能够到达那一个城市的纸

②如果我们在**最远处**范围内找不到纸价更低的城市，我们就看能不能直接到家，如果可以，就买**刚好能够直接到家**的纸

③在②的基础上，如果不能到家，我们就将纸**买满**，然后到**我们能够一次性到达**的城市里面**纸价最低**的城市

可以证明，这样我们所消耗的**每一张纸**的纸价都是最低的

重复上述过程直至抵达终点，算法结束

代码：

```
#include<stdio.h>
#define maxn 1000
int len[maxn],val[maxn];
char s[maxn];
int n,k;
int find_cheaper(int now);
int find_next(int now);
int can_back_home(int now);
int main(){
    //freopen("3.in","r",stdin);
    //freopen("std.out","w",stdout);
    int t;
    scanf("%d",&t);
    while(t--){
        int flag=1;
    //scanf("%s",s);
    scanf("%d%d",&n,&k);
    int p,d;
    for(int i=0;i<=n;i++){
        scanf("%d%d%d",&val[i],&p,&d);
        len[i]=p*d;
    }
    for(int i=0;i<=n;i++){
        if(len[i]>k){
            printf("-1\n");
            flag=0;
```

```
            break;
        }
    }
    if(!flag)continue;
    int now=0,rest=0;
    long long ans=0;
    while(now!=n+1){
        int next=find_cheaper(now);
        if(next){
            int path=0;
            for(int i=now;i<next;i++){
                path+=len[i];
            }
            ans+=(path-rest)*val[now];
            rest=0;
            now=next;
            continue;
        }
        next=can_back_home(now);
        if(next){
            int path=0;
            for(int i=now;i<=n;i++){
                path+=len[i];
            }
            ans+=(path-rest)*val[now];
            now=n+1;
            break;
        }
        next=find_next(now);
        ans+=(k-rest)*val[now];
        rest=k;
        for(int i=now;i<next;i++){
            rest-=len[i];
        }
        now=next;
    }
    printf("%lld\n",ans);
    }
    return 0;
}

int find_cheaper(int now){
    int tot=0,next=now;{
        for(int i=now+1;i<=n;i++){
            if(tot+len[i-1]<=k){
                if(val[i]<val[now]){
                    return i;
                }
                tot+=len[i-1];
            }
            else{
                break;
            }
        }
    }
    return 0;
}
```

```
int can_back_home(int now){
    int tot=0;
    for(int i=now;i<=n;i++){
        tot+=len[i];
    }
    if(tot<k){
        return 1;
    }
    else{
        return 0;
    }
}

int find_next(int now){
    int place=now+1,minumval=val[now+1],tot=0;
    for(int i=now+1;i<=n;i++){
        if(tot+len[i-1]<=k){
            if(val[i]<minumval){
                minumval=val[i];
                place=i;
            }
            tot+=len[i-1];
        }
        else{
            break;
        }
    }
    return place;
}
```