DIFFICULTY:

MEDIUM

PRE-REQUISITES:

Dynamic Programming

PROBLEM:

Given interview time lengths and departure times of trains of **K(<101)** candidates, find the minimum number of candidates that'll miss their trains if we optimally schedule the interviews. Any interview cannot be paused in between.

Assume that to catch a train, the interview of each candidate has to end at least 30 minutes before the departure time of the train the candidate wishes to take.

Some candidates have trains next day, so for them -1 is given as departure time. Also, since departure times are given in minutes it will be less than 24*60.

EXPLANATION:

We use dynamic programming. First we'll sort candidates according to decreasing order of departure times and start processing from last candidate in our dp. We define **F(i, startTime)** as the answer for candidates **1,2,...i**, if we start scheduling interviews at time **startTime**.

Note that we'll ignore the candidates with departure time -1 as they don't contribute to answer and we can always schedule there interviews after all other interviews.

Here will be the recurrences:

```
//i'th candidate misses the train
//no need to schedule the interview
//because it'll lead to time wastage
F(i,startTime)= 1 + F(i-1, startTime)

//schedule the interview of i'th candidate
//if he/she doesn't miss the train
if(startTime+30+interview_time[i] <= departure_time[i])
   F(i,startTime)= min(F(i,startTime), F(i-1,startTime + interview_time[i]))
```

)

Complexity: **O(K*24*60)** worst case.

Code :

```
#include<bits/stdc++.h>
using namespace std;

int dp[109][24*70];
pair < int, int > cand[109];

int rec(int i, int curtime){
    //base case
    if(i<0)return 0;

    int &ret=dp[i][curtime];

    if(cand[i].first==-1)return ret=rec(i-1,curtime);

    //dp already calculated
    if(ret!=-1)return ret;

    //don't assign for interview: the i'th candidate
    ret=1+rec(i-1,curtime);

    //assign for interview i'th candidate iff possible
    if(curtime+cand[i].second+30<=cand[i].first)
        ret=min(ret, rec(i-1,curtime+cand[i].second));

    return ret;
}

int main(){
    int p;
    cin >> p;
    for(int ii=1; ii<=p; ii++){
        memset(dp,-1,sizeof(dp));
        int n;
        cin >> n;

        for(int i=0; i<n; i++)
            cin >> cand[i].second;
```

```
        for(int i=0; i<n; i++)
            cin >> cand[i].first;

        //sort on increasing basis of departure times
        sort(cand,cand+n);

        //reverse, because we'll process the candidate in increasing order of departure
times
        reverse(cand,cand+n);

        cout << "Case " << ii << ": " << rec(n-1,0) << endl;
    }
    return 0;
}
```