няколко начина за получаване на максимален брой ябълки, програмата трябва да изведе най-малкия възможен номер на последната достигната площадка.

ПРИМЕР

Вход	**Изход**
4 2 2 3 7 8 9 9 0	16 2

Решение

Разглеждаме множеството от подзадачи $\{T_i, i=0,1,\ldots,N\}$ подобни на дадената, при които търсим максималния брой ябълки, които Скокльо може да събере, тръгвайки от началото и завършвайки в площадката на i-тия метър. Подзадачата T_i може да се реши, ако вече имаме решенията на подзадачите за по-малки стойности на i. Тази идея на динамичното оптимиране е реализирана в приложената програма. В `p[i]` въвеждаме броя ябълки, намиращи се на `i`-тата площадка, а позволените дължини на скоковете са в елементите на масива `s[j]`. В `t[i]` пресмятаме последователно решенията на описаните по-горе подзадачи. Очевидно `t[0]=p[0]`, а стойността на `t[i]`, за `i`>0 е равна на `p[i]` плюс най-голяма от тези стойности на `t[k]`, които са такива, че от позиция `k` може да се стигне с един скок до позиция `i`. Решението на основната задача е най-голямата стойност на масива `t[i]`.

```
#include <iostream>
using namespace std;
const int N=200000, M=200;
int n,m; int s[M]; int p[N], t[N];
int main()
{
   cin >> n >> m;
   for(int j=1;j<=m;j++) cin >> s[j];
   for(int i=0;i<=n;i++) cin >> p[i];
   t[0]=p[0];
   for(int i=1;i<=n;i++)
   {
      int v=-1;
      for(int j=1;j<=m;j++)
      {  int k=i-s[j];
         if(k>=0) if(t[k]>-1) if(v<t[k]) v=t[k];
      }
      if(v>-1) t[i]=v+p[i]; else t[i]=-1;
   }
   int max=t[0]; int pos=0;
   for(int i=1;i<=n;i++)
   if(t[i]>max) {max=t[i]; pos=i;}
   cout << max << " " << pos << endl;
}
```

Задача B3. MUSIC IDOL, автор Зорница Дженкова

Млади хора от цялата страна участвали в поредния кастинг на Music Idol. Някои кандидати имали музикален опит, други били завършили различни

школи, но всички са участвали в предварителни изпити и всеки има получен бал, който се изразява с дробно число от 0 до 100, с точност до 4 цифри след десетичната точка. Журито изслушвало участниците от град Варна в продължение на N дни, $1 < N < 200$. За да има ред, организаторите на кастинга направили така, че участниците да пристигат в града само през нощта преди съответния работен ден на журито (включително и в нощта преди първия работен ден) и се настанявали в един хотел. Всеки ден, журито избирало да прослушва определен брой от настанените в хотела участници, които имали най-висок бал, а ако имало такива с равен бал, от тях се избирали тези, които имали по-ранна регистрация в хотела. След като един участник бъдел прослушан, той веднага трябвало да напусне хотела. Хотелът имал 10000 места и никога не се препълнил по време на целия кастинг. След N–тия ден в хотела имало останали неизслушани участници, а в нощта преди (N+1)–вия работен нови кандидати не били настанявани.

Напишете програма **`MUSIC`**, която намира името на първия неизслушан участник, т.е. този, който би трябвало да бъде изслушан първи през (N +1)-вия ден, ако журито би решило да работи един ден повече от предварително определените дни.

Вход

Данните се четат от стандартния вход. На първия ред е записано числото N. Следват N реда, всеки започващ с броя M на пристигналите участници през нощта преди съответния ден, $0 \le M < 500$, следван от M двойки, съставени от името на поредния участник и неговия бал. Накрая на реда е записан броят K на действително изслушаните от журито участници за деня, $0 \le K < 300$. Всички числа и имена са разделени с по един интервал. Имената са съставени от по най-много 10 малки и главни латински букви.

ПРИМЕР

Вход	**Изход**
3 3 Ana 90 Krum 90 Kalina 99 2 0 0 1 Roza 90.0001 1	Krum

Решение

Данните за всеки участник – името, балът и поредният номер на регистрация в хотела – са представени в структурата `data`. За решаване на задачата е използвана приоритетна опашка Q, реализирана в STL. След прочитане на данните, отнасящи се за поредния ден, те се внасят в Q, в съответствие с указаните приоритети – първо по намаляваща стойност на бала, а при равен бал – по нарастващ номер на пристигане в хотела. Приоритетът на данните в опашката се определя от класа `cmp`, където освен големината на бала, се взема предвид и поредността на регистрацията в хотела. След това се изваждат от Q тези участници, които са били прослушани за същия ден. Този процес се повтаря N пъти, колкото са дните на конкурса. Нкрая се извежда името на този участник, които ще се окаже на върха на приоритетната опашка.

```
#include<iostream>
#include<queue>
using namespace std;
struct data
{ string s; double b; int r; };
struct cmp:
public binary_function<data, data, bool>
{ bool operator()(data x, data y)
   {if(x.b < y.b) return true;
    if(x.b==y.b) if(x.r > y.r) return true;
    return false;
   }
};
priority_queue<data, vector<data>, cmp> Q;
int main()
{int c=0;
 int n; cin >> n;
 for(int i=1;i<=n;i++)
 { int m; cin >> m;
   for(int j=1;j<=m;j++)
    { data d;
      cin >> d.s >> d.b;
      d.r=c; c++;
      Q.push(d);
    }
   int k; cin >> k;
   for(int j=1; j<=k; j++) Q.pop();
 }
 cout << (Q.top()).s << endl;
}
```


Група С

Задача С1. РЕДИЦИ, автор Емил Келеведжиев

Да разгледаме редицата (3,1,4,2,4). Тя може да бъде разбита на две подредици – (3,4) и (1,2,4), които са строго растящи, т.е. елементите им са сортирани в строго нарастващ ред. *Разбиване* означава, че всяка от двете подредици трябва има поне по един елемент, обединението на елементите на двете подредици трябва да се състои от елементите на дадената редица и никой елемент от дадената редица не трябва едновременно да е елемент и на двете подредици. Забележете, че за редицата (4,8,1,5,3) такова разбиване не може да се направи. Напишете програма **SEQ**, която "познава" дали дадена редица от цели положителни числа може да бъде разбита на две строго растящи подредици.

Вход

На първия ред на стандартния вход е зададен броят N на редиците, които програмата трябва да провери, $N \leq 10$. Всеки от следващите N реда дефинира по една редица. Редът започва с броя M на елементите в съответната редица, $2 \leq M \leq 100\,000$. В същия ред, след един интервал, са зададени елементите на редицата, разделени един от друг с интервали. Всички елементи са цели положителни числа, не надминаващи 1 милиард.

Изход

На стандартния изход програмата трябва да изведе низ с дължина N, съставен от нули и единици (без разделящи интервали), с резултатите от направените проверки. За поредната редица низът трябва да съдържа цифрата 1, ако е възможно тази редица да се разбие на две строго растящи подредици, или цифрата 0, ако такова разбиване не е възможно.

ПРИМЕР

Вход	**Изход**
2 5 3 1 4 2 4 5 4 8 1 5 3	10

Решение

Наивната идея да образуваме всички възможни разбивания на редицата в две подредици и да проверяваме, дали получените подредици са строго растящи, ще доведе до алгоритъм с експоненциално нарастване на броя на стъпките, при увеличаване размерите на задачата. Алгоритъм с линейна сложност се получава по следния начин. Да допуснем, че успешно сме разбили началото (съставено от първите I елемента) на редицата и a и b са последните (максималните) елементи на двете подредици. За началото, съставено от първия елемент, такова разбиване е възможно и единствено – в едната подредица е първият елемент, а втората е празна. Максималният измежду a и b да означим с max. Да разгледаме (I+1)-вия елемент c. Ако $c > max$, трябва да го причислим към подредицата на max. Ако $c < max$ – опитваме да го причислим към другата подредица. Ако това е възможно – получили сме разбиване на първите I+1 елемента. Ако с добавянето му там строгата монотоннос на подредицата се нарушава, ясно е, че задачата няма решение.