```
void gen(int k)
{ if(k==30) show();
  int min,max;
  if(k==0) min=0;  else min=c[k-1];
  if(k<12) max=84; else max=210;
    for(c[k]=min; c[k]<max; c[k]++)
    if(ok(k)) gen(k+1);
}
int main()
{ init();
  gen(0);
}
```

Група В

Задача В1. СУМИ, автор Младен Манев

Напишете програма **SUM**, която да намира броя на начините, по които зададено цяло положително число *N* може да се представи като сума на поне две последователни положителни цели числа. Числото 5, например, може да се представи като такава сума точно по един начин: 5=2+3. Числото 8, пък, изобщо не може да се представи така (0 начини), а числото 21 има три такива представяния: 21=10+11=6+7+8=1+2+3+4+5+6.

Вход

Програмата трябва да въвежда цяло число *N* ($2 < N < 10^8$) от стандартния вход.

Изход

Програмата трябва да изведе на стандартния изход броя на начините, по които ***N*** може да се запише като сума на поне две последователни положителни цели числа.

ПРИМЕР

Вход	**Изход**
9	2

Решение

Нека n е такова, че може да се представи като сума на k последователни положителни цели числа, най-малкото от които е m. Тогава

$$n = m + (m+1) + ... + (m+k-1) = \frac{m + (m+k-1)}{2} \cdot k .$$

Следователно броят на търсените начини е равен на броя на наредените двойки от цели числа (m,k), които са решения на диофантовото уравнение $2n = (2m+k-1)k$ и за които $m > 0$, $k > 1$. Тъй като $2m = \frac{2n}{k} - k + 1$ и $k < 2m + k - 1$, то наредената двойка числа (m,k) ще е решение на задачата, ако:

- k дели $2n$;
- $\frac{2n}{k} - k + 1$ е четно число;
- $k^2 < 2n$.


```
#include<iostream>
using namespace std;
int main()
{  int n;
   cin >> n;
   int s=0;
   for(int k=2;k*k<2*n;k++)
      if (!(2*n%k) && !((2*n/k-k+1)%2)) s++;
   cout << s << endl;
   return 0;
}
```

Задача В2. СКОКОВЕ, автор Емил Келеведжиев

По дължината на една алея, на разстояние един метър една от друга, са разположени *N*+1 площадки, номерирани с целите числа от 0 до *N*, където *N* е цяло число, $1 < N < 200\,000$. На всяка площадка е поставена кошница с зададен брой ябълки. Скокльо е известен със способността си да извършва *M* различни по дължина (измерена в метри) скока, $0 < M < 200$. Скокльо обича да се разхожда по алеята, като скача от площадка на площадка и събира ябълките които намери в съответната кошница. Разходките му винаги започват от началото на алеята (площадката с номер 0). Когато е на някоя площадка, той може да скочи, с един от възможните *M* скока, на някои от следващите площадки (такава с по-голям номер). Разходката може да бъде прекратена на произволна площадка до която Скокльо е достигнал.

Скокльо може да взема ябълки само от кошниците на площадките, в които е попаднал, включително от тази, от която е тръгнал и тази, в която е прекратил разходката. Напишете програма **SKOK**, която да му помогне да събере максимален брой ябълки.

Вход

Програмата трябва да прочете входните данни от стандартния вход. На първия ред са зададени числата *N* и *M*, разделени с интервал. На втория ред са зададени *M* числа – позволените дължини на скоковете в метри. Всяка от тези дължини е цяло положително число, по-малко от 1000. На третия ред са зададени *N*+1 цели числа – бройките на ябълките, подредени в нарастващ ред на номерата на съответните им площадки. На една площадка може да има най-малко нула и най-много 1000 ябълки. Всички числа във втория и третия ред на входа са разделени с по един интервал.

Изход

Програмата трябва да изведе на един ред в стандартния изход две цели числа, разделени с един интервал. Първото от тези числа трябва да е равно на максималния брой ябълки, които Скокльо може да събере по време на разходката, а второто число да е равно на номера на последната площадка, до която той е достигнал, получавайки този максимален брой. Ако са възможни

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.