```
   int l = 0, r = 0;
   string inp;
   for (int i = 0; i < n; ++i)
   {
      cin >> inp;
      if (inp == "L") ++l;
      if (inp == "R") ++r;
      if (inp == "P")
      {
         cout << min(l, r) << " ";
         if (l > r)
         {
            cout << "L" << endl;
            l -= r;
         }
         else
         {
            cout << "R" << endl;
            r -= l;
         }
      }
  }
  cout << l << endl << r << endl;
  return 0;
}
```

Задача D2. ЧИСЛО НА ЕДИНГТЪН, автор Красимир Манев

Всеки уважаващ себе си велосипедист трябва да знае собственото си *число на Единтън*, а то е най-голямото цяло неотрицателно E такова, че поне E пъти в живота си е изминавал с велосипеда си поне E метра без почивка. Естествено, ако един човек никога не е карал велосипед, числото му на Единтън е нула. Но младият програмист Станчо е и запален велосипедист. Научавайки за числото на Единтън той решил да пресметне, колко е то в неговия случай. Записал на един лист броя N на случаите, в които е бил на седлото на любимия си „байк" без почивка и по колко метра е изминал във всеки от тях. Започнал да пресмята, но числата били толкова много, че скоро се забъркал в сметките. Знанията му по програмиране, пък, не били достатъчни и затова оставя на Вас да се справите с проблема. Напишете програма **EDI**, която да пресмята числото на Единтън.

Вход

На първия ред на стандартния вход ще бъде зададено числото N, $1 \le N \le 100000$. Всеки от следващите N реда ще съдържа по едно цяло положително число L, дължината в метри измината при поредното каране $1 \le L \le 10000$.

Изход

На стандартния изход програмата трябва да изведе намереното число на Единтън.

ПРИМЕР

Вход	Изход
10 3 4 12 5 2 6 2 8 1 7	5

Решение

Това, че числото N е голямо, е без значение. Решението на задачата, което предлагаме, ще може да работи и с много по-големи стойности на N – милиони и даже милиарди. Важното е, че при всяко каране Станчо е изминавал не повече от 10000 метра. Затова дефинираме масив `int a[10001]` и даваме стойност нула на елементите му (първият цикъл `for`). В него ще натрупаме статистиката за каранията, които е осъществил Станчо – в `a[i]` ще намерим колко пъти е изминавал по `i` метра (вторият цикъл `for`). Числото на Единтън намираме в променливата `E` с цикъла `while`. Първата стойност, която проверяваме е 10000. За да има Станчо число на Единтън 10000 е необходимо `a[10000]`≥10000. Ако това е така, изпълнението на цикъла ще се прекрати и програмата ще изведе 10000. Ако това не е така, за да има Станчо число на Единтън 9999 е необходимо `a[9999]+a[10000]`≥9999. Затова, в тялото на цикъла, към съдържанието на `a[E-1]` (в случая `a[9999]`) добавяме съдържанието на `a[E]` (в случая `a[10000]`) и намаляваме стойността на `E` (след `E--` в променливата `E` ще имаме 9999). Така ще накараме оператора `while` да сравни броя на каранията от поне 9999 метра с 9999 и т.н. На всяка стъпка на цикъла, при която `E` още не е търсеното число на Единтън, ще добавяме общия брой на каранията от поне `E` метра към броя на тези от `E-1` метра, за да намерим броя на каранията от поне `E-1` метра и да го сравним с `E-1`. Тъй като дължините са положителни числа, натрупваната сума ще расте, а числото в `E` ще намалява и неравенството `a[E]<E`, рано или късно, ще бъде нарушено. Изпълнението на цикъла ще се прекрати и съдържанието на `E` ще бъде търсеното число на Единтън.

```
#include <iostream>
using namespace std;
int main()
{
   int a[10001],N,i,L,E;
   cin>>N;
   for(i=0;i<=10000;i++) a[i]=0;
```

```
   for(i=1;i<=N;i++)
   {  cin >> L; a[L]++; }
   E=10000;
   while(a[E]<E) a[E-1]+=a[E--];
   cout << E << endln;
   return 0;
}
```

Друг начин да се реши задачата, не толкова бърз и рационален при използването на памет, е да се съхранят данните в масив с максимум 100000 елемента и да се сортират елементите му в намаляващ ред. Отговорът е най-големият индекс, за който стойността на масива е по-голяма или равна на индекса.

```
#include <cstdio>
#include <vector>
#include <algorithm>
#include <functional>

using namespace std;

int main()

{
   int n;
   vector<int> dist;
   scanf("%d",&n);
   dist.resize(n);
   for (int i=0;i<n;++i)
      scanf("%d",&dist[i]);
   sort(dist.begin(),dist.end(),greater<int>());
   for (int i= 0;i<n && dist[i]>=i+1;++i);
   printf("%d\n", i);
   return 0;
}
```

Бързината на алгоритъма за сортиране е от съществено значение за това решение на задачата. Предложеното решение използва вградения QSORT. Всъщност, натрупването на статистика в първото решиние също е вид сортиране – чрез броене. То е възможно, защото броят на различните възможни числа е ограничен до 10000.

Задача D3. БАНКОВИ СМЕТКИ, автор Венета Богданова

Напоследък в пресата се срещат съобщения, че личните данни на гражданите не са добре защитени. На първокурсниците от специалност "Приложна математика" в един университет поставили за домашно да измислят начин да кодират номерата на банкови сметки IBAN. Росица измислила едно правило, но то е доста трудно за прилагане. Идеята е следната:

- Номерът на сметка IBAN е последователност от 22 знака – главни английски букви и цифри. Например BGFINV1234KL0987BC1234.
- Всеки знак от номера IBAN се заменя с двойка цифри. Пред цифрите от 1 до 9 се добавя нула, а латинските букви се кодират по следния начин 'A' с 10, 'B' с 11, 'C' с 12, 'D' с 13 и т.н. до 'Z' с 35, с изключение на буквата 'O' – тя, както и цифрата '0', се кодира с две нули.

A	B	C	D	E	F	G	H	I	J	K	L	M
10	11	12	13	14	15	16	17	18	19	20	21	22
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
23	00	25	26	27	28	29	30	31	32	33	34	35

Така номерът BGFINV1234KL0987BC1234 се преобразува до
111615182331010203042021000908071112010203 04

- Полученото в стъпка 2 число се умножава с 11 и това е окончателно кодираната сметка. Т.е. за горния пример се получава:
  1227767005641112233462231009988782232112233 44.

Помогнете на Росица в затруднението. Напишете програма **IBAN**, която да извършва описаното кодиране на номера на банкови сметки.

Вход

На единствения ред на стандартния вход ще бъде зададен номер на банкова сметка IBAN. Той ще се състои от 22 знака – главни английски букви и цифри.

Взход

На стандартния изход програмата трябва да изведе кодираният номер на сметката.

ПРИМЕР 1

Вход

```
BGFINV1234KL0987BC1234
```

Изход

```
1227767005641112233462231009988782232112233 44
```

ПРИМЕР 2

Вход

```
0TTTTTTTTTTTTTTTTTTTT
```

Изход

```
00322222222222222222222222222222222222222219
```

Решение

И двете предложени решения извършват една и съща последователност от действия, но използват различни структури от данни. Прочита се дадената банкова сметка. Замества се всеки знак с два знака по описаното правило. Умножението по 11 всъщност е събиране на полученото число и същото число, с добавена една нула в края, т.е. десет пъти по-голямо от първото.

Вариант 1:

```
#include<iostream>
#include<fstream>
#include<string.h>
using namespace std;
```