Državno takmičenje iz programiranja za učenike osnovnih škola, 21.05.2017.

Vaša rešenja možete testirati u takmičarskom okruženju, na platformi Petlje,
http://petlja.org/Competitions/Competition/29

1. **[V, VI] Kralj**

vremensko ograničenje 0.5 s memorijsko ograničenje 64 MB

Na šahovskoj tabli 8×8 nalazi se samo kralj. Kralj se u svakom potezu može pomeriti jedno polje i to na bilo koje susedno polje onom polju na kojem stoji, u bilo kom od 8 pravaca (pravo, ali i dijagonalno). Napisati program koji izračunava na koliko se polja kralj može pomeriti ako kreće sa nekog datog polja. Program treba da ispiše broj poteza kralja za svako od tri učitana polazna polja (ta tri slučaja su nezavisna).

Ulaz

Na standardnom ulazu su u tri reda date tri polazne pozicije kralja. Pozicije su date pomoću dva karaktera, u formatu **kv** gde **k** je obeležje kolone (malo slovo engleske abecede od a do h), **v** je obeležje vrste (cifra od 1 do 8).

Izlaz

Na standarni izlaz ispisati tri cela broja, svaki u posebnom redu koji predstavljaju broj poteza kralja sa svake od tri učitane pozicije.

Primer

Ulaz

a8
b2
h3

Izlaz

3
8
5

Sa polja a8, kralj može da se pomeri na polja a7, b8 i b7, sa polja b2 na polja a1, a2, a3, b1, b3, c1, c2 i c3, dok sa polja h3 može da se pomeri na polja h2, h4, g2, g3 i g4.

Rešenje

Postoje ukupno 3 rešenja ovog problema.
Ako se kralj nalazi na poljima a1, a8, h1, h8, onda se u narednom potezu može pomeriti na jedno od 3 susedna polja.
Ako se kralj nalazi na ivicama table (na poljima a2,a3,a4,a5,a6,a7, h2,h3,h4,h5,h6,h7, b1,c1,d1,e1,f1,g1,e1, b8,c8,d8,e8,f8,g8,e8), onda se u narednom potezu može pomeriti na jedno od 5 susednih polja.
Inače, odgovor je 8.

```
#include <iostream>
#include <string>

using namespace std;

int main() {
  for (int i = 0; i < 3; i++) {
```

```
    string s;
    cin >> s;
    char k = s[0];
    char v = s[1];
    bool kolona_na_ivici = k == 'a' || k == 'h';
    bool vrsta_na_ivici  = v == '1' || v == '8';
    int broj_polja;
    if (kolona_na_ivici && vrsta_na_ivici)
      broj_polja = 3;
    else if (kolona_na_ivici || vrsta_na_ivici)
      broj_polja = 5;
    else
      broj_polja = 8;
    cout << broj_polja << endl;
  }
  return 0;
}
```

2. [V, VI] Arapski u rimski

vremensko ograničenje 1 s memorijsko ograničenje 64 MB

Rimski brojevi se zapisuju pomoću slova I, V, X, L, C, D, M. Na primer, broj 15 se zapisuje kao XV, a broj 148 kao CXLVIII. Napiši program koji prevodi uneti arapski broj u rimski.

Ulaz
Sa ulaza se unosi prirodan broj n ($1 \leq n \leq 2000$)
Izlaz
Na standardni izlaz se ispisuje rimski zapis broja n zapisan velikim slovima latinice.

Primer 1
Ulaz
1978
Izlaz
MCMLXXVIII

Primer 2
Ulaz
2000
Izlaz
MM

Rešenje
Rimski zapis koristi različite simbole za zapis cifara jedinica, cifara desetica i cifara stotica, ali je princip zapisa svake od deset dekadnih cifara od tih simbola isti. I za jedinice i za desetice i za stotine koristi se po tri karakteristična simbola: simbol s_1 koji predstavlja vrednost 1, simbol s_5 koji predstavlja vrednost 5 i simbol s_{10} koji predstavlja vrednost 10. U slučaju cifre jedinice to su simboli I, V i X, u slučaju cifre desetice to su simboli X, L i C, a u slučaju cifre stotine to su simboli C, D i M.
Cifra 0 se zapisuje praznom niskom, cifra 1 jednim simbolom s_1, cifra 2 pomoću dva simbola s1, cifra 3 pomoću tri simbola s_1, cifra 4 simbolima s1s5,
cifra 5 simbolom s5, cifra 6 simbolima s5s1, cifra 7 simbolima s5s1s1, cifra 8 simbolima s5s1s1s1, a cifra 9 simbolima s_1s_{10}. Zato definišemo posebnu funkciju koja za datu

cifru i data tri simbola s_1, s_5 i s_{10} od tih simbola gradi nisku koja predstavlja zapis te cifre. Ona može izvršiti klasifikaciju kom intervalu pripada data cifra c (konstrukcijom `else-if`, slično kao u zadatku [Agregatno stanje](../../../02 Grananje/04 Slozeno grananje/02 intervali/01 agregatno_stanje)) i ako je ona manja od 4 vratiti nisku dobijenu tako što se c ponovi simbol s_1, ako je jednaka 4 vratiti nisku s_1s_5, ako je manja od 9 vratiti nisku koja se dobije kada se na simbol s_5 simbol s_1 nadoveže c-5 puta, a ako je jednaka 9 vratiti nisku s_1s_{10}. U jeziku C++ se niska koja sadrži karakter `c` `k` puta gradi konstruktorom `string(k, c)`, a u jeziku C# konstruktorom `new string(c, k)`.

Glavna funkcija konverzije izdvaja cifru po cifru broja n operacijom celobrojnog ostatka i uklanja je operacijom celobrojnog količnika pri deljenju sa 10, krenuvši od cifre jedinica (slično kao u zadatku [Zbir cifara](../../../01 Aritmetika/02 Celobrojno deljenje/04 Pozicioni zapis/01 Brojevi/01 zbir_cifara)). Svaka tako dobijena cifra se konvertuje u rimski zapis pomoćnom funkcijom, prosleđujući joj simbole s_1, s_5i s_{10} u zavisnosti od težine tekuće cifre i taj zapis se dodaje ispred dosadašnjeg izgrađenog rezultata (koji kreće od prazne niske i proširuje se na levo). Postupak se završava kada se obrade sve cifre broja n tj. kada se on celobrojnim deljenjem svede na nulu. Ako se to ne desi ni nakon uklanjanja cifre stotine, preostali broj (to je broj hiljada) se konvertuje u rimski tako što se taj broj puta na početak rezultata doda simbol M.

```
#include <iostream>
#include <string>

using namespace std;

// zapis jedne rimske cifre - simbol1, simbol5 i simbol 10 odredjuju
// da li se radi i cifri jedinica, desetica ili stotina
string rimskaCifra(int c, char simbol1, char simbol5, char simbol10) {
  if (c < 4) // npr. "", "I", "II", "III"
    return string(c, simbol1);
  else if (c == 4) // npr. "IV"
    return string(1, simbol1) + string(1, simbol5);
  else if (c < 9) // npr. "V", "VI", "VII", "VIII"
    return string(1, simbol5) + string(c-5, simbol1);
  else // npr. "IX"
    return string(1, simbol1) + string(1, simbol10);
}

// prevodi dati broj u rimski zapis
string arapskiURimski(int n) {
  string rezultat = "";
  // cifra jedinica
  rezultat = rimskaCifra(n % 10, 'I', 'V', 'X');
  n /= 10; // uklanjamo je
  // ako nema vise cifara vracamo rezultat
  if (n == 0) return rezultat;
  // cifra desetica
  rezultat = rimskaCifra(n % 10, 'X', 'L', 'C') + rezultat;
  n /= 10; // uklanjamo je
  // ako nema vise cifara vracamo rezultat
  if (n == 0) return rezultat;
  // cifra stotina
  rezultat = rimskaCifra(n % 10, 'C', 'D', 'M') + rezultat;
```

```
  n /= 10; // uklanjamo je
  // ako nema vise cifara vracamo rezultat
  if (n == 0) return rezultat;
  // dodajemo hiljade na pocetak rezultata i vracamo ga
  return string(n, 'M') + rezultat;
}


int main() {
  // ucitavamo broj
  int n;
  cin >> n;
  // prevodimo ga u rimski zapis i ispisujemo rezultat
  cout << arapskiURimski(n) << endl;
  return 0;
}
```

3. **[V, VI] Najkraća dopuna do palindroma**

vremensko ograničenje	memorijsko ograničenje
0.1 s	64 MB

Niska abaca nije palindrom (ne čita se isto sleva i sdesna), ali ako joj na početak dopišemo karaktere ac, dobijamo nisku acabaca koja jeste palindrom (čita se isto i sleva i sdesna). Napiši program koji za datu nisku određuje dužinu najkraćeg palindroma koji se može dobiti dopisivanjem karaktera s leve strane date niske.

Ulaz
Sa standardnog ulaza se unosi niska sastavljena samo od N ($1 \leq N \leq 50000$ malih slova engleske abecede.
Izlaz
Na standardni izlaz se ispisuje tražena dužina najkraće proširene niske koja je palindrom.

Primer 1
Ulaz
abaca
Izlaz
7
Primer 2
Ulaz
abcdefg
Izlaz
13
Rešenje je gfedcbabcdefg.

Primer 3
Ulaz
anavolimilovana
Izlaz
15
Reč je već palindrom, pa se ne dopisuje ni jedan karakter i rešenje je anavolimilovana.

Primer 4

Ulaz
anavolimilovanakapak
Izlaz
25
Rešenje je kapakanavolimilovanakapak.

Rešenje

Sporije rešenje (pokriva vremensko ograničenje za 24 test primera)

Neka je data ulazna niska s="abbac"

Dovoljno je da postavimo poslednje slovo c ispred sadržaja niske s i dobićemo nisku "cabbac", koje jeste palindrom s minimalnom dužinom, i zadovoljava uslove zadatka.

Sličan postupak se može primeniti i u opštem slučaju, kada ulazna niska *s* sadrži podniz na početku niske koji je palindrom.

Predstavimo ulaznu nisku u obliku s = s1 + s2, gde s1 je najduži prefix od s, koji je palindrom.

Tada možemo da sastavimo niz t = s3 + s1 + s2, gde niska s3 je dobijena obrtanjem niske s2 (pročitane kao u ogledalu) .

Kako niska s1 je najduži palindrom, onda niska t je palindrom, napravljen po uslovima zadatka i to najkraći moguć.

Dužina niske t je:

L = n2+n1+n2=2*n2 + n1,

gde n1 je dužina za s1, n2 je dužina za s2 (naravno i za s3). Polazna dužina niske s je n tj. n = n1 + n2. Dakle, L = 2*(n-n1) + n1 = 2*n – n1 i program mora da ispiše L.

Ako želino da odredimo n1, onda redom proveravamo sve prefikse niske s i tražimo najduži palindrom među njima. Njegova dužina je n1.

```
#include <iostream>
#include <string>

using namespace std;

bool jePalindrom(const string& s, int n) {
  for (int i = 0, j = n - 1; i < j; i++, j--)
    if (s[i] != s[j])
      return false;
  return true;
}

int duzinaNajduzegPalindromskogPrefiksa(const string& s) {
  for (int n = s.size(); n >= 1; n--)
    if (jePalindrom(s, n))
      return n;
}

int main() {
  string s;
  cin >> s;
  // s razlazemo na prefiks + sufiks tako da je prefiks sto duzi
```

```
  // palindrom. Tada je trazeni palindrom (palindrom koji se dobija sa
  // sto manje dopisivanja slova na pocetak niske s) jednak:
  // obrnut_sufiks + prefiks + sufiks
  int duzinaPrefiksa = duzinaNajduzegPalindromskogPrefiksa(s);
  int duzinaSufiksa = s.size() - duzinaPrefiksa;
  int duzinaNajkracegPalindroma = duzinaSufiksa + duzinaPrefiksa + duzinaSufiksa;
  cout << duzinaNajkracegPalindroma << endl;
  return 0;
}
```

Algoritam složenosti O(n)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

int duzinaNajduzegPalindromskogPrefiksa(const string& s) {
  string sObratno(s.rbegin(), s.rend());
  string str = s + '.' + sObratno;
  vector<int> kmp(str.size() + 1, 0);
  kmp[0] = -1;
  for (int i = 0; i < str.size(); i++) {
    int k = i;
    while (k > 0 && str[i] != str[kmp[k]])
      k = kmp[k];
    kmp[i + 1] = kmp[k] + 1;
  }
  return min(kmp[kmp.size() - 1], (int)s.size());
}

int main() {
  string s;
  cin >> s;
  // s razlazemo na prefiks + sufiks tako da je prefiks sto duzi
  // palindrom. Tada je trazeni palindrom dobijen sa sto manje
  // dopisivanja slova na pocetak jednak:
  //    obrni(sufiks) + prefiks + sufiks
  int duzinaPrefiksa = duzinaNajduzegPalindromskogPrefiksa(s);
  int duzinaSufiksa = s.size() - duzinaPrefiksa;
  int duzinaNajkracegPalindroma = duzinaSufiksa + duzinaPrefiksa + duzinaSufiksa;
  cout << duzinaNajkracegPalindroma << endl;
  return 0;
}
```

4. [V, VI] Broj kolone u tabeli

vremensko ograničenje	memorijsko ograničenje
0.5 s	64 MB

U programima za tabelarna izračunavanja (Microsoft Excel, LibreOffice Calc i slično) kolone su obeležene slovima i to kao A, B, ..., Z, AA, AB, ..., AZ, BA, BB, ..., ZZ, AAA, ... Napiši program koji omogućava konverziju tekstualne oznake kolone u redni broj kolone (od 1 pa naviše).

Ulaz
Sa standardnog ulaza se unosi tekstualna oznaka kolone koja sadrži velika slova engleske abecede (njih najviše 5).
Izlaz
Na standardni izlaz ispisati broj koji odgovara toj koloni.

Primer 1
Ulaz
D
Izlaz
4

Primer 2
Ulaz
AB
Izlaz
28

Primer 3
Ulaz
ZZZZZ
Izlaz
12356630

Rešenje

```
#include <iostream>
#include <string>

using namespace std;

int main() {
  string nazivKolone;
  cin >> nazivKolone;
  int brojKolone = 0;
  for (char c : nazivKolone)
    brojKolone = brojKolone * 26 + (c - 'A' + 1);
  cout << brojKolone << endl;
  return 0;
}
```

Kategorija 2

1. [VII, VIII] Stolice

vremensko ograničenje	memorijsko ograničenje
0.5 s	64 MB

Na jednom kvadratnom metru svečane sale možemo da postavimo ili jedan sto kvadratnog oblika ili jednu stolicu. Oko svakog stola može da se postavi najviše 8 stolica (po jedna uz svaku stranu i po jedna uz svako teme kvadrata). Napiši program koji će izračunati najveći broj stolica u sali, ako nijednu stolicu nije moguće postaviti bez pripadajućeg stola.

Ulaz
U jedinom redu standardnog ulaza dati su dužina svečane sale R i širina svečane sale K razdvojeni razmakom. Obe dimenzije R i K su izražene u celom broju metara i važi 0<R,K<1000.
Izlaz
Na standardni izlaz ispiši jedan ceo broj koji predstavlja najveći mogući broj stolica.

Primer 1
Ulaz
3 4
Izlaz
10

Jedan mogući raspored stolica i stolova je sledeći (stolice su obeležene znakom +, a stolovi znakom o).
++++
+o+o
++++
Raspored
++++
+o++
++++
nije ispravan jer stolice u poslednjem redu nemaju pridružen sto.

Primer 2
Ulaz
5 5
Izlaz
21

Primer 3
Ulaz
100 101
Izlaz
8944

Rešenje
```
#include <iostream>
using namespace std;
int main() {
  // broj redova i kolona
  int r, k;
```

```
  cin >> r >> k;

  int brojStolova;
  if (r % 3 == 0 && k % 3 == 0)
    brojStolova = (r / 3) * (k / 3);
  else if (r % 3 == 0 && k % 3 != 0)
    brojStolova = (r / 3) * (k / 3) + (r / 3);
  else if (r % 3 != 0 && k % 3 == 0)
    brojStolova = (r / 3) * (k / 3) + (k / 3);
  else
    brojStolova = (r / 3) * (k / 3) + (k / 3) + (r / 3) + 1;

  int brojStolica = r * k - brojStolova;
  cout << brojStolica << endl;

  return 0;
}
```

2. [VII, VIII] Rimski u arapski

vremensko ograničenje	memorijsko ograničenje
0.5 s	64 MB

Rimski brojevi se zapisuju pomoću slova I, V, X, L, C, D, M. Na primer, broj 15 se zapisuje kao XV, a broj 148 kao CXLVIII. Napiši program koji konvertuje uneti rimski broj u arapski broj.

Ulaz
Jedina linija standardnog ulaza sadrži jedan ispravno zapisan rimski broj iz intervala od 1 do 2000.

Izlaz
Na standardni izlaz ispisati uneti broj zapisan u uobičajenom (arapskom) zapisu.

Primer 1
Ulaz
MCMLXXVIII
Izlaz
1978

Primer 2
Ulaz
MMXVII
Izlaz
2017

Rešenje

Ovaj zadatak je dualan zadatku [Arapski u rimski](../10 arapski_u_rimski), ali im se rešenja prilično razlikuju. Vrednost ispravno zapisanog rimskog broja se može odrediti tako što čitamo njegove cifre s leva na desno i na tekući zbir dodeljujemo vrednost svake cifre (M ima vrednost 1000, D vrednost 500, C vredost 100, L vrednost 50, X vrednost 10, V vrednost 5 i I vrednost 1).
Jedini izuzetak od ovog pravila je slučaj kada se u zapisu pojavi manja ispred veće cifre i tada se od tekućeg zbira ta cifra oduzima. Na primer, izračunajmo vrednost zapisa MCMLXXIV. Na osnovu

prethodne diskusije, vrednost ovog broja je 1000 - 100 + 1000 + 50 + 10 + 10 - 1+ 5 tj. 1974. Rezultat se inicijalizuje na nulu. Prva cifra je M iza koje nije veća cifra, pa se rezultat uvećava za 1000. Nakon toga nastupa slučaj u kojem sledi cifra C koja je manja od naredne cifre M, tako da se od zbira oduzima vrednost 100 i dobija se vrednost 900. Nakon toga se na zbir dodaje 50, pa dva puta 10 i dobija se zbir 1970, dok se ne naiđe na cifru I koja je manja od njoj naredne cifre V. Zato se od zbira oduzima vrednost 1 i dobija se 1969, pre nego što se na kraju doda vrednost poslednje cifre V i dobije konačan rezultat 1974.

Prolaz kroz sve rimske cifre datog rimskog broja vršimo u sklopu jedne petlje u kojoj indeksna promenljiva `i` menja svoju vrednost od nule, pa sve dok je manja od dužine niske kojom je zapisa redni broj. U telu petlje se za svaku cifru proverava da li neposredno iza nje postoji cifra koja je od nje veća (pri tom se mora voditi računa i o tome da tekuća cifra nije poslednja). Ako takva cifra postoji, onda se tekući zbira umanji za vrednost tekuće cifre, u suprotnom se na zbir dodaje vrednost tekuće cifre. U svakom slučaju indeksna promenljiva se uvećava za 1 (preskače se tekuća i prelazi se na narednu cifru, ako ona postoji).

Vrednosti cifara možemo čuvati u posebnom asocijativnom nizu (mapi, rečniku) tj. možemo u jeziku C++ upotrebiti bibliotečku strukturu podataka `map` a u jeziku C# `Dictionary`. Druga mogućnost je definisanje pomoćne funkcije koja određuje vrednost date cifre i to grananjem (najbolje naredbom `switch`, koju smo prikazali, na primer, u zadatku [Broj dana u mesecu](../../../02 Grananje/04 Slozeno grananje/02 intervali/08 broj_dana_u_mesecu)).

```
#include <iostream>
#include <string>
#include <map>

using namespace std;

int rimskiUArapski(string rimski) {
  // vrednost svake rimske cifre
  map<char, int> cifre = { {'M', 1000}, {'D',  500}, {'C', 100}, {'L', 50}, {'X', 10}, {'V', 5}, {'I', 1} };
  // rezultat - vrednost rimski zapisanog broja
  int arapski = 0;
  // prolazimo kroz sve cifre rimskog broja
  for (int i = 0; i < rimski.size(); i++) {
    // ako neposredno iza tekuce postoji cifra veca od nje
    if (i + 1 < rimski.size() && cifre[rimski[i]] < cifre[rimski[i+1]])
      // rezultat umanjujemo za vrednost tekuce cifre
      arapski -= cifre[rimski[i]];
    else
      // inace rezultat uvecavamo za vrednost tekuce cifre
      arapski += cifre[rimski[i]];
  }
  // vracamo konacan rezultat
  return arapski;
}

int main() {
  // ucitavamo rimski zapis broja
  string rimski;
  cin >> rimski;
  // odredjujemo mu vrednost i ispisujemo je
```

```
  cout << rimskiUArapski(rimski) << endl;
  return 0;
}
```

3. [VII, VIII] Pikado

vremensko ograničenje	memorijsko ograničenje
1.3 s	64 MB

Kružna meta koju koristimo u specijalnom pikadu izdeljena je u koncentrične kružne prstenove. Širina prstenova može biti međusobno različita. Napiši program koji za datu tačku određuje zonu kojoj pripada.

Ulaz
Sa standardnog ulaza unosi se broj n ($1 \leq n \leq 50000$), a zatim i n realnih brojeva zaokruženih na dve decimale, svaki u posebnom redu, koji predstavljaju širine svih kružnih prstenova. Nakon toga se unosi broj m ($1 \leq m \leq 50000$) i zatim m parova koordinata tačaka (u svakom redu se nalaze dva realna broja zaokružena na dve decimale, razdvojena sa po jednim razmakom).

Izlaz
Na standardni izlaz ispisati m linija. U svakoj liniji ispisati ili indeks zone (broje se od nule) kojoj tačka pripada ili tekst izvan ako je tačka izvan poslednje zone. Ako je tačka na granici dve zone, smatrati da pripada unutrašnjoj.

Primer

Ulaz
3
3.0
1.0
6.0
3
2.0 0.0
18.0 20.0
4.0 4.0

Izlaz
0
izvan
2

Rešenje

Tačka pripada nekoj zoni ako i samo ako je njeno rastojanje od koordinatnog početka (a njega možemo izračunati korišćenjem Pitagorine teoreme, kao u zadatku [Rastojanje tačaka](../../../../01 Aritmetika/01 Formule/01 Poznate formule/04 rastojanje_tacaka)) strogo veće od unutrašnjeg, a manje ili jednako spoljašnjem prečniku te zone. Unutrašnji prečnik neke zone jednak je zbiru širina svih zona pre nje, ne uključujući njenu širinu, a spoljašnji prečnik jednak je zbiru širina svih zona pre nje, uključujući i njenu širinu. Primetimo da je unutrašnji prečnik neke zone jednak spoljašnjem prečniku prethodne zone (osim kod zone 0, koja ima unurašnji prečnik nula). Zato je dovoljno da izračunamo spoljašnje prečnike svih zona, a oni se mogu izračunati kao parcijalni zbirovi širina zona. Njih možemo računati

inkrementalno, dodajući svaku novu učitanu širinu na tekući zbir (slično kao što smo u zadatku [Prefiks najvećeg zbira](../../../../03 Petlje/01 serije_brojeva/08 rekurentne_serije/06 prefiks_najveceg_zbira) računali zbirove svih prefiksa) ili, u jeziku C++ bibliotečkom funkcijom `partial_sum`. Kada je poznat niz spoljašnjih prečnika zona zona u kojem se tačka nalazi se može naći tako što se nađe prvi spoljašnji prečnik zone koji je veći ili jednak rastojanju tačke od koordinatnog početka. To možemo uraditi binarnom
pretragom, bilo ručno implementiranom (kao, na primer, u zadatku [Prvi veći](../02 prvi_veci_poslednji_manji)), bilo pomoću bibliotečkih funkcija `lower_bound` u jeziku C++ ili `BinarySearch` u jeziku C#.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <cmath>

using namespace std;

int main() {
  ios_base::sync_with_stdio(false);

  // ucitavamo niz sirina zona
  int n;
  cin >> n;
  vector<double> zone(n);
  for (int i = 0; i < n; i++)
    cin >> zone[i];

  // izracunavamo poluprecnike zona (kao zbir sirina svih zona zakljucno sa tom)
  partial_sum(zone.begin(), zone.end(), zone.begin());

  // ucitavamo m tacaka
  int m;
  cin >> m;
  for (int i = 0; i < m; i++) {
    // ucitavamo koordinate tacke (x, y)
    double x, y;
    cin >> x >> y;
    // rastojanje tacke (x, y) od koordinatnog pocetka
    double r = sqrt(x*x + y*y);

    // pronalazimo prvu zonu takvu da joj je poluprecnik veci ili jednak r
    auto it = lower_bound(zone.begin(), zone.end(), r);

    // ako takva ne postoji, tacka je izvan svih zona
    if (it == zone.end())
      cout << "izvan" << endl;
    else
      cout << distance(zone.begin(), it) << endl;
  }
  return 0;
}
```

4. [VII, VIII] Kolona u tabeli sa datim brojem

vremensko ograničenje	memorijsko ograničenje
0.5 s	64 MB

U programima za tabelarna izračunavanja (Microsoft Excel, LibreOffice Calc i slično) kolone su obeležene slovima i to kao A, B, ..., Z, AA, AB, ..., AZ, BA, BB, ..., ZZ, AAA, ... Napiši program koji omogućava konverziju rednog broja kolone (od 1 pa naviše) u tekstualnu oznaku kolone.

Ulaz
Sa standardnog ulaza se unosi broj n ($1 \leq n \leq 10$), a zatim n brojeva (svaki u posebnom redu) iz intervala od 1 do 12356630.

Izlaz
Na standardni izlaz ispisati n stringova (svaki u posebnom redu) koji predstavljaju oznake kolona koje odgovaraju unetim brojevima.

Primer

Ulaz
3
18
100
1000000
Izlaz
R
CV
BDWGN

Rešenje

Malo ćemo modifikovati algoritam određivanja cifara u datoj brojevnoj osnovi koji, podsetimo se, određuje poslednju cifru operacijom ostatka pri deljenju brojevnom osnovom i uklanja je operacijom celobrojnog deljenja (isto kao, na primer, u zadatku [Zbir cifara](../../../01 Aritmetika/02 Celobrojno deljenje/04 Pozicioni zapis/01 Brojevi/01 zbir_cifara) ili
[Broj i zbir cifara broja](../../../03 Petlje/01 serije_brojeva/09 cifre_broja/01 broj_i_zbir_cifara_broja)).

Ako se od broja oduzme 1, poslednja cifra je između 0 i 25 i može se odrediti određivanjem ostatka pri deljenju brojem 26. Ako se na taj ostatak doda kôd karaktera `'A'`, dobija se kod poslednjeg karaktera u slovnoj oznaci kolone.

Recimo da je izgradnja string uzastopnim dodavanjem karaktera na njegov početak generalno veoma neefikasna operacija (složenost joj je kvadratna). Ipak, pošto se u zadatku radi sa veoma kratkim stringovima, nećemo obraćati pažnju na to.

```
#include <iostream>
#include <string>

using namespace std;

string odBroja(int broj) {
  string rez = "";
```

```
  do {
    broj--;
    rez = string(1, 'A' + broj % 26) + rez;
    broj /= 26;
  } while (broj > 0);
  return rez;
}

int main() {
  int m;
  cin >> m;
  for (int i = 0; i < m; i++) {
    int broj;
    cin >> broj;
    cout << odBroja(broj) << endl;
  }
  return 0;
}
```