U svim zadacima memorijsko ograničenje 64MB, vremensko ograničenje 2s

1. Data su dva cela broja *a, b* (0<a<b<1 000 000 000 000 000) i ceo broj *c* (1<c<1 000 000 000 000 000). Napisati C/C++ program DELJIVOST koji će ispisati koliko celih brojeva između *a* i *b* (uključujući i *a* i *b*) je deljivo datim brojem *c* . U prvom redu standardnog ulaza dat je prirodan broj *a*, u drugom redu standardnog ulaza dat je prirodan broj *b*, u trećem redu standardnog ulaza dat je prirodan broj *c*. U jedinom redu standardnog izlaza ispisati traženi broj sadržalaca.
PRIMER

ULAZ	IZLAZ	ULAZ	IZLAZ	ULAZ	IZLAZ
8	7	8	5	8	154320986
40		40		1234567890	
5		8		8	

REŠENJE

Naivno rešenje: isprobati proveru deljivosti brojem C za sve brojeve između A i B.

ALI,
kada ne bi bilo postavljeno dozvoljeno vreme za rad programa, onda jedno od rešenja bi bilo isprobati proveru deljivosti brojem C za sve brojeve između A i B. No, postoji opasnost da prekoračimo vremensko ograničenje, jer broj tih provera može biti veoma velik za sve brojeve čija vrednost prelazi sto miliona.

Da bi se rešio ovaj problem sa manjim brojem provera i time zadovoljilo vremensko ograničenje, primećujemo da brojevi između A i B, koji su deljivi brojem C, se javljaju u redovnim intervalima, od kojih svaki sadrži skup uzastopnih C brojeva.

Koliko ima takvih intervala? Broj tih intervala odgovara broju sadržalaca broja C koji su između brojeva A i B.

Ostaje još da se pažljivo nađe najmanji i najveći sadržalac broja C koji su između brojeva A i B odnosno brojevi a1 i b1, te je ukupan rezultat jednak 1+(b1-a1)/c (gde a1 je najmanji sadrzalac broja c, ne manji od broja a, b1 je najveci sadrzalac broja c, ne veci od broja b).

```
#include<iostream>
using namespace std;

long long int a,b,c;

int main()
{
 cin >> a >> b >> c;
 long long int a1=a;
 if(a%c!=0) a1=a+c-a%c;  // najmanji sadrzalac od c, ne manji od broja a
 long long int b1=b-b%c; // najveci sadrzalac broja c, ne veci od broja b
 long long int r=0;
 if(a1<=b1) r = 1+(b1-a1)/c; //formula za prebrojavanje
 cout << r << endl;
}
```

2. Za razliku od Euklidske geometrije, Hermann Minkowski je u 19. veku istraživao Menheten (taxicab) geometriju. U toj geometriji udaljenost između dve tačke T1(x1,y1), T2(x2,y2) definiše se kao:
D(T1,T2) = |x1 - x2| + |y1 - y2|
Sve ostale definicije jednake su kao i u običnoj geometriji, pa tako i definicije kružnice i kruga koje glase: Kružnica je skup tačaka jednako udaljenih od neke tačke (koju nazivamo centar kružnice). Krug je skup tačaka ravni omeđen kružnicom. Napisati C/C++ program MENHETN koji računa površinu kruga u

geometriji Minkowskog i euklidskoj geometriji.
U prvom i jedinom redu standardnog ulaza nalazi se poluprečnik R, prirodni broj manji od ili jednak 10000. U prvi red standardnog izlaza ispišite površinu kruga poluprečnika R u euklidskoj geometriji (na 6 decimala). U drugi red treba ispisati površinu kruga poluprečnika R u geometriji Minkowskog (na 6 decimala). Dopušteno odstupanje od preciznog rešenja je ±0.0001.

ULAZ	IZLAZ
21	1385.442360
	882.000000

REŠENJA

Krug u Menheten (taxicab) geometriji je u kvadrat u euklidskoj geometriji. Povrsina kruga u Minkowski geometriji je $(\text{sqrt}(2)*r)^2 = 2* r^2$.

Povrsina kruga u Euklidskoj geometriji je $r^2 * pi$.
Pi mozemo zapisati kao konstantu ili je izracunati kao 4*atan(1) ili acos(-1).
Neke implementacije jezika C definišu konstantu M_PI u zaglavlju <math.h>, iako konstanta nije definisana u C standardu.

Većina C++ kompilatora podržava korišćenje konstante ***M_PI*** (po uzoru na programski jezik C), međutim, da bi se ta konstanta mogla upotrebiti, kod nekih kompilatora potrebno je na početku programa (pre ***include*** direktive) navesti definiciju
#define _USE_MATH_DEFINES

i tek nakon nje uključiti zaglavlje ***cmath***. Napominjemo da je navedenu definiciju u programu potrebno navesti pre svih uključivanja zaglavlja, jer se može desiti i da neko drugo zaglavlje indirektno uključi ***cmath***.

Drugi način je definisanje konstante u programu
const double PI = 3.14159265359;

Ali, postavlja se pitanje broja decimala koje je potrebno navesti (dve, tri decimale koje učenici znaju napamet često nisu dovoljne).

Alternativa je da se pi izračuna korišćenjem nekih trigonometrijskih funkcija (npr. kao što smo gore naveli 4*atan(1) ili acos(-1)).

```
#include <math.h>
#include <stdio.h>

using namespace std;

int main() {
   float r;
   scanf("%f", &r);
   printf("%.6f\n", r*r*M_PI);
   printf("%.6f", 2*r*r);
   return 0;
}
```

3. Svake godine učenici imaju priliku da u školskim svečanim salama prisustvuju proslavi školske slave. Razredne starešine su odlučile da rasporede učenike prema prozivniku na sledeći način: najpre se u sali popunjava prvi red u smeru slevo nadesno, potom na isti način se popunjava drugi red i tako redom dok se ne popuni cela sala koja ima n redova sa po m mesta za sedenje u svakom redu. Međutim, kad je došao direktor škole, obrazložio je razrednim starešinama zašto takav raspored sedenja nije korektan i predložio je

preraspodelu sedenja na sledeći način: u svakom redu (od prvog do poslednjeg) učenici treba najpre da popune sva prva mesta, potom sva druga mesta i tako dalje (popunjava se sala po kolonama). Napišite program RASPORED, kojim se unose celi brojevi n, m ($1 < m < 3*10^4$, $1 < n < 3*10^4$) i koji izračunava i ispisuje koliko učenika u preraspodeli će ostati na svojim prvobitnim mestima. Smatrajte da škola ima dovoljno učenika da popuni celu salu.

PRIMER 1

ULAZ	**IZLAZ**
3 3	3

PRIMER 2

ULAZ	**IZLAZ**
2 4	2

Objašnjenje za prvi primer: ako n = 3 i m = 3, prvobitni raspored učenika je

1 2 3
4 5 6
7 8 9

Nakon preraspodele, raspored učenika je:

1 4 7
2 5 8
3 6 9

Učenici s rednim brojevima 1, 5 i 9 će ostati na svojim mestima.

REŠENJE

```
#include <iostream>
using namespace std;

int nzd(int a,int b)
{
  while (b)
  {    int r = a % b;
       a = b;
      b = r;
  }
  return a;
}

int main()
{  int n, m;
  cin>>n>>m;
  cout<<1+nzd(n - 1, m - 1) << endl;
  return 0;
}
```

Čitljivije i sporije rešenje

Naivan pristup zahteva da proveravamo pozicije u polaznoj tabeli i transponovanoj (“okrenutoj”) tabeli. Problem ovog pristupa je što za relativno veliko m*n proveravamo sve pozicije. Iako znamo da neke pozicije uopšte ne bi morale da budu kandidati za proveru.

```
#include <iostream>
using namespace std;
int main()
{
  long int n, m;
```

```
 cin>>n>>m;
long int r = 0;

 for (int z=0;z<n*m;z++)
  if (z / n == z % m && z / m == z % n)
    ++r;

 cout<<r<<endl;
 return 0;
}
```

4. Napisati C/C++ program NIVEN koji učitava sa standardnog ulaza prirodan broj n<=1000 i na standardni izlaz ispisuje prvih n Nivenovih brojeva u istom redu razdvojenih sa po jednim blanko karakterom. Prirodan broj je Nivenov ako je deljiv sumom svojih cifara.

ULAZ	IZLAZ
15	1 2 3 4 5 6 7 8 9 10 12 18 20 21 24

REŠENJE

```
#include <iostream>
using namespace std;

int n,pom,zbir;

int main()
{
   ios::sync_with_stdio(false);
   cin>>n;
   for(int i=1,nadjeni=0;nadjeni<n;i++)
   {
      pom=i,zbir=0;

      //racunamo zbir cifara za visecifreni broj pom
      while(pom>0)
      {
         zbir+=pom%10,pom=pom/10;
      }

     //provera da li je broj Nivenov
      if(i%zbir==0)
      {
         cout<<i<<" ";
         nadjeni++;
      }
   }
}
```