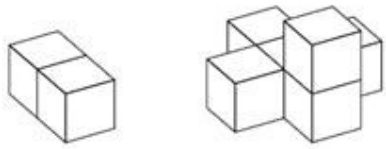
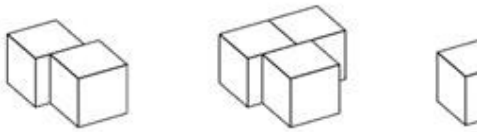
Zadatak 1 – Actečka piramida (2 sec, 256MB)

Actečke piramide su izgrađene od kamenih blokova. Svaki blok je kocka stranice 1. Actečki kralj postavlja prvi blok na zemlju, a zatim se dodaju ostali blokovi tako da svaki sljedeći blok ima zajedničku stranu sa bar jednim od prethodno postavljenih blokova.

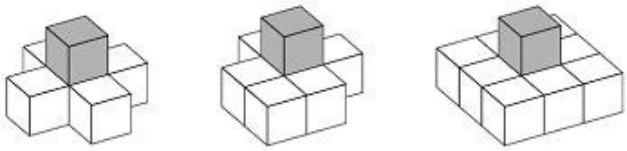


Ispravan raspored blokova

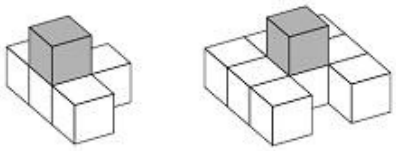


Neispravan raspored blokova

Blok A je stabilan ako stoji na zemlji ili ako stoji na drugom bloku B, pri čemu je svaka strana bloka B oslonjena na drugi blok ili na zemlju. Da bi preživjela vjekove, cijela piramida mora biti stabilna tj. svaki blok u njoj mora biti stabilan.



Stabilni blokovi



Nestabilni blokovi

Kralj Acteka vas je unajmio da izračunate visinu najviše moguće stabilne piramide koja se može izgraditi od raspoloživih blokova.

Ulaz: Jedini red ulaza sadrži jedan cio broj n – broj raspoloživh blokova, uključujući i prvi koji postavlja kralj. ($1 \leq n \leq 10^9$).

Izlaz: Štampati jedan cio broj – visinu najviše moguće stabilne piramide sastavljene od n blokova.

Test primjeri

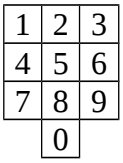
Ulaz	Izlaz
6	2
5	1
20	3

Rješenje:

```
#include <cstdio>
#include <iostream>

using namespace std;

int main()
{
    int n = 0, h = 0;
    cin >> n;
    for (int l = 1, c = 1; c <= n; h++, l += h * 4, c += l);
    cout<<h;

  return 0;
}
```

Zadatak 2 – Stanari (1 sec, 256 MB)

Na ulazu u jednu stambenu zgradu postavljena je mala numerička tastatura (kao na slici) koju stanari koriste za otvaranje vrata zgrade. Da bi bilo koji stanar ušao u zgradu, on mora najprije unijeti (ukucati) svoj lični tajni broj (LTB) koji se sastoji od pet cifara. Kada računar prepozna da je unijet ispravan LTB, vrata se otvaraju.



Kako su tasteri dosta blizu jedan drugog, često se događa da stanari prilikom unošenja LTB-a griješe pa ga moraju unositi više puta. Kućni savjet je na sastanku donio odluku da se izmijeni program koji otvara vrata tako da prihvaća i LTB koji je "skoro ispravan". Preciznije, stanari žele da sistem otvori vrata i onda kada prilikom ukucavanja LTB-a stanar pogriješi tako da **jednom** umjesto željenog tastera pritisne neki njemu susjedni taster. Susjednim smatramo one tastere koji se nalaze neposredno iznad, ispod, lijevo ili desno (vidi sliku). Na primjer, susjedi tastera označenog brojem 4 su 1, 5 i 7.

Napišite program koji će određivati da li neki petocifreni broj otvara vrata po ovim novim pravilima. Data je lista ispravnih LTB-ova i lista kombinacija koje su unesene na tastaturi. Vaš program treba za svaku unesenu kombinaciju odredi da li je prihvaćena ili odbijena, i ako je prihvaćena štampa kojem LTB-u odgovara. Pritom, ako je unesena kombinacija jednaka nekom LTB-u treba štampati taj isti LTB, a inače treba štampati prvi LTB iz liste kojem kombinacija odgovara.

Ulaz: U prvom redu ulaza nalazi se prirodan broj N, $1 \le N \le 10$, broj datih LTB-ova. U svakom od sljedećih N redova nalazi se po jedan LTB. U sljedećem redu nalazi se prirodni broj M, $1 \le M \le 10$, broj unijetih kombinacija. U svakom od sljedećih M redova nalazi se po jedna kombinacija. Svaki LTB i svaka kombinacija su niz od tačno pet cifara.

Izlaz: Izlaz se sastoji od M redova: u i-ti red treba štampati rezultate unošenja i-te kombinacije. Ako kombinacija otvara vrata, potrebno je štampati odgovarajući LTB, a inače riječ **POGRESNO**.

Test primjeri

Ulaz	**Izlaz**
5 12345 77777 22234 52234 76324 3 12645 52234 77788	12345 52234 POGRESNO

Rješenje:

Simulacija postupka opisanog u tekstu zadatka.

```
#include <stdio.h>

#define MAXN 100
#define MAXM 100
#define FIN "ltb.in"  // ako ucitavamo iz fajla
```

```
#define FOUT "ltb.out" // ako stampamo u fajl
#define KRIVO "POGRESNO"
const susjedi[10][10] =
{  {  0, 0, 0, 0, 0, 0, 0, 0, 1, 0 },
  {  0, 0, 1, 0, 1, 0, 0, 0, 0, 0 },
  {  0, 1, 0, 1, 0, 1, 0, 0, 0, 0 },
  {  0, 0, 1, 0, 0, 0, 1, 0, 0, 0 },
  {  0, 1, 0, 0, 0, 1, 0, 1, 0, 0 },
  {  0, 0, 1, 0, 1, 0, 1, 0, 1, 0 },
  {  0, 0, 0, 1, 0, 1, 0, 0, 0, 1 },
  {  0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },
  {  1, 0, 0, 0, 0, 1, 0, 1, 0, 1 },
  {  0, 0, 0, 0, 0, 0, 1, 0, 1, 0 } };

long pravi[MAXN], upisani[MAXM], rjesenje[MAXM];
int np, nu;

void ulaz(void)
{
  int i;
  FILE *f = fopen(FIN, "rt");
  fscanf(f, "%d", &np);
  for (i=0; i<np; i++)
    fscanf(f, "%ld", &pravi[i]);
  fscanf(f, "%d", &nu);
  for (i=0; i<nu; i++)
    fscanf(f, "%ld", &upisani[i]);
  fclose(f);
}

int slicni(long a, long b)
{
  int z, susjednih;
  susjednih = 0;
  for (z=0; z<5; z++)
  {
    if (a%10 != b%10)
      if (!susjedi[a%10][b%10])
      {
        susjednih = 2;
        break;
      }
      else
      {
        susjednih++;
        if (susjednih > 1)
          break;
```

```
      }
    a /= 10;
    b /= 10;
  }
  return susjednih<=1;
}

void rijesi(void)
{
  int u, p;
  for (u=0; u<nu; u++)
  {
    rjesenje[u] = -1;
    for (p=0; p<np; p++)
      if (upisani[u]==pravi[p])
      {
        rjesenje[u] = pravi[p];
        break;
      }
    if (rjesenje[u]!=-1) continue;
    for (p=0; p<np; p++)
      if (slicni(upisani[u],pravi[p]))
      {
        rjesenje[u] = pravi[p];
        break;
      }
  }
}

void izlaz(void)
{
  int i;
  FILE *f = fopen(FOUT, "wt");
  for (i=0; i<nu; i++)
    if (rjesenje[i]!=-1)
      fprintf(f, "%ld\n", rjesenje[i]);
    else
      fprintf(f, "%s\n", KRIVO);
  fclose(f);
}

int main()
{
  ulaz();
  rijesi();
  izlaz();
  return 0;
```

```
}
```

Zadatak 3 – Par (2 sec, 256MB)

Za date prirodne brojeve a i b odrediti brojeve x i y, takve da je NZS(a, b) = NZS(x, y), HZD(a, b) = NZD(x, y) i y − x je minimalno. NZS je najmanji zajednički sadržalac a NZD je najveći zajednički djelilac.

Ulaz: U prvom redu ulaza su dva broja a i b ($1 \leq a, b \leq 10^9$).

Izlaz: Štampati dva prirodna broja x i y ($1 \leq x \leq y$) koja zadovoljavaju uslove zadatka.

Test primjeri

Ulaz	Izlaz
3 4	3 4
1 12	3 4

Rješenje:

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

#ifdef WIN32
    #define LLD "%I64d"
#else
    #define LLD "%lld"
#endif

template <typename T>
T abs(T x) {
    return x > 0 ? x : -x;
}

int gcd(int a, int b) {
    while (b) {
        a %= b;
        std::swap(a, b);
    }
    return a;
}

long long lcm(int a, int b) {
    return 1LL * a / gcd(a, b) * b;
}

long long power(long long x, int p) {
    long long ans = 1;
    for (int i = 0; i < p; i++)
        ans *= x;
    return ans;
}

void factorize(long long number, std::vector <long long> &primes) {
    primes.clear();
    for (long long i = 2; i * i <= number; ++i) {
        int cnt = 0;
        while (!(number % i))
            ++cnt, number /= i;
        if (cnt)
            primes.push_back(power(i, cnt));
    }
```

```
    if (number > 1)
        primes.push_back(number);
}

long long bestAns = 0, bestX, bestY;
long long x = 1, y = 1;

void go(int pos, std::vector <long long> &primes) {
    if (pos == (int)primes.size()) {
        if (bestAns > abs(x - y)) {
            bestAns = abs(x - y);
            bestX = x;
            bestY = y;
        }
        return;
    }

    x *= primes[pos];
    go(pos + 1, primes);
    x /= primes[pos];

    y *= primes[pos];
    go(pos + 1, primes);
    y /= primes[pos];
}

int main() {
    freopen("gcm.in", "r", stdin);
    freopen("gcm.out", "w", stdout);
    int a, b;
    scanf("%d%d", &a, &b);

    int d = gcd(a, b);
    std::vector <long long> primes;
    factorize(lcm(a, b) / d, primes);

    bestAns = abs(b - a) / d;
    bestX = a / d, bestY = b / d;
    go(0, primes);

    if (bestX > bestY)
        std::swap(bestX, bestY);
    printf(LLD" "LLD"\n", bestX * d, bestY * d);
    return 0;
}
```