Zadaci sa rešenjima Kvalifikacije II 2011.

zadatak: Operacija

Data su četiri broja *A*, *B*, *C* i *op*. Vrednost *op* ima sledeće značenje:

- *op* = 1 označava sabiranje,
- *op* = 2 označava oduzimanje,
- *op* = 3 označava množenje,
- *op* = 4 označava deljenje.

Za date ulazne vrednosti ispisati 1 ako *A op B = C*, inače ispisati 0.
Ulaz.
(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom redu standardnog ulaza nalaze se prirodni brojevi *A*, *B*, *C* i *op* ($0 \leq A, B, C \leq 10{,}000{,}000{,}000$, $1 \leq op \leq 4$).
Izlaz.
(Izlazne podatke ispisati na standardan izlaz.) U prvi i jedini red standardnog izlaza ispisati traženu vrednost.
Ograničenja.
U bar 50% test primera će *op* imati vrednost 1 ili 2.
60% test primera će imati vrednost *A*, *B* i *C* iz intervala [0, 100].
Primer 1.

standardni ulaz	standardni izlaz
1 2 3 1	1

Objašnjenje.
Izraz 1 + 2 = 3 je tačan, te je izlaz 1.
Primer 2.

standardni ulaz	standardni izlaz
2 1 2 4	1

Primer 3.

standardni ulaz	standardni izlaz
3 2 1 4	0

Objašnjenje.
3 / 2 = 1.5, te je izraz 3 / 2 = 1 netačan.
Primer 4.

standardni ulaz	standardni izlaz
4 5 21 3	0

Objašnjenje.
4 * 5 = 20.
Primer 5.

standardni ulaz	standardni izlaz
55 5 50 2	1

fajl: operacija.cpp

```
#include <iostream>
#include <cstdio>
#include <fstream>
#include <vector>
#include <cmath>
#define ffor(_a,_f,_t) for(int _a=(_f),__t=(_t);_a<__t;_a++)
#define all(_v) (_v).begin() , (_v).end()
#define sz size()
#define pb push_back
#define SET(__set, val) memset(__set, val, sizeof(__set))
#define FOR(__i, __n) ffor (__i, 0, __n)

using namespace std;

const int MAXN = 100001;
```

```
int a[MAXN];

char str[MAXN + 10];

int main(){

  long long A, B, C;
  int op;
  scanf("%lld %lld %lld %d", &A, &B, &C, &op);

  bool good = false;
  if (op == 1){
    if (A + B == C)
      good = true;
  }
  else if (op == 2){
    if (A - B == C)
      good = true;
  }
  else if (op == 3){
    if (B == 0 && C == 0)
      good = true;
    if (B != 0){
      op = 4;
      swap(A, C); // turn the problem into div
    }
  }
  if (op == 4){
    if (B != 0 && A % B == 0 && A / B == C)
      good = true;
  }

  if (good)
    printf("1\n");
  else
    printf("0\n");
  return 0;
}
```

zadatak: Plus-minus

Dat je niz znakova $(s_1, ..., s_n)$ dužine n. Znak s_i je + ili -. Dat je niz od $n + 1$ brojeva $(a_1, ..., a_{n+1})$. Treba rasporediti brojeve u dati niz znakova tako da vrednost dobijenog izraz bude maksimalna. Formalno, treba naći vrednost *val* koja je definisana na sledeći način:

$$val = \max\{a_{p(1)}\ s_1\ ...\ a_{p(n)}\ s_n\ a_{p(n+1)} \mid p \text{ je permutacija brojeva od 1 do } n + 1\}$$

Ulaz.
(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom redu standardnog ulaza nalazi se prirodan broj n ($1 \le n \le 100000$). U drugom redu nalazi se n znakova, redom od s_1 do s_n. Svaki znak je karakter '+' ili '-'. Znakovi nisu odvojeni razmakom. U trećem redu se nalazi $n + 1$ brojeva, brojevi od a_1 do a_{n+1}, odvojenih razmakom. Svaki od tih brojeva je iz intervala [0, 1000000].

Izlaz.
(Izlazne podatke ispisati na standardan izlaz.) U prvi i jedini red standardnog izlaza ispisati traženu vrednost, odnosno vrednost *val*.

Primer 1.

standardni ulaz	**standardni izlaz**
3 +-+ 1 2 3 4	8

Objašnjenje.
Jedno rešenje predstavlja raspored brojeva
2+3-1+4

Primer 2.

standardni ulaz	**standardni izlaz**
2	6

```
++
1 3 2
```
Objašnjenje.
Bilo koji raspored brojeva vodi ka optimalnom rešenju.
Primer 3.

standardni ulaz	**standardni izlaz**
4	6

```
----
3 12 1 2 0
```
Objašnjenje.
Bilo koji raspored takav da je na prvom mestu broj 12 vodi ka optimalnom rešenju.

fajl: plusminus.cpp

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <fstream>
#include <vector>
#include <cmath>
#define ffor(_a,_f,_t) for(int _a=(_f),__t=(_t);_a<__t;_a++)
#define all(_v) (_v).begin() , (_v).end()
#define sz size()
#define pb push_back
#define SET(__set, val) memset(__set, val, sizeof(__set))
#define FOR(__i, __n) ffor (__i, 0, __n)

using namespace std;

const int MAXN = 100001;

int a[MAXN];

char str[MAXN + 10];

int main(){
  int n;
  scanf("%d", &n);
  scanf("%s", str);
  FOR (i, n + 1)
    scanf("%d", &a[i]);

  int cnt = 0;
  FOR (i, n)
    cnt += str[i] == '-';
  sort(a, a + n + 1);
  long long ret = 0LL;
  FOR (i, n + 1)
    if (i < cnt)
      ret -= a[i];
    else
      ret += a[i];

  printf("%lld\n", ret);
  return 0;
}
```

zadatak: Kvadratici

Data je celobrojna kvadratna rešetka u ravni dimenzija *n* x *m* (ima *nm* kvadratića). Koliko ima različitih kvadrata čija su sva temena u čvorovima ove rešetke (stranice tih kvadrata ne moraju da budu paralelne sa ivicama kvadratne rešetke)?

Ulaz.
(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom i jedinom redu standradnog ulaza nalaze se dva prirodna broja n i m - dimenzije kvadratne rešetke ($1 \leq n, m \leq 10^9$).
Izlaz.
(Izlazne podatke ispisati na standardan izlaz.) Neka je traženi broj kvadrata K. Na standardni izlaz ispisati **ostatak** pri deljenju broja K sa $10^9 + 7$.
Ograničenja.
U 40% test primera $1 \leq n, m \leq 100$.
U 60% test primera $1 \leq n, m \leq 1.000$.
U 80% test primera $1 \leq n, m \leq 1.000.000$.
Primer 1.

standardni ulaz	standardni izlaz
2 3	10

Primer 2.

standardni ulaz	standardni izlaz
500 501	271062715

fajl: kvadratici.cpp

```
#include <cstdlib>
#include <cstdio>
#include <cmath>

const int MODUO = 1000000007;

long long n, m, sol, tmp1, tmp2;

int main() {

  scanf("%d%d", &n, &m);
  if (m < n) {
    long long tmp = m; m = n; n = tmp;
  }

  sol = 0;

  tmp1 = ((n + 1) * (m + 1)) % MODUO;
  tmp2 = (n * (n + 1) / 2) % MODUO;
  sol = (sol + tmp1 * tmp2) % MODUO;

  sol = (sol + tmp2 * tmp2) % MODUO;

  tmp1 = ((n + m + 2) * (2 * n + 1)) % MODUO;
  tmp1 = (tmp1 * tmp2) % MODUO;
  sol = (sol - tmp1 * 333333336) % MODUO;
  // ovo je umesto deljenja sa 3 jer je 333333336 inverz za 1/3
  // u polju Z_p, gde je p = 10^9 + 7 (prost broj)

  //bitno
  if (sol < 0) sol += MODUO;

  printf("%d\n", sol);
  return 0;
}
```

fajl: kvadratici.pas

```
const
  MODUO = 1000000007;

var
  m, n, sol, tmp : int64;
  i : longint;
```

```
begin

  readln(n, m);

  if (m < n) then begin
      tmp := m; m := n; n := tmp;
  end;

  sol := 0;

  for i := 1 to n do begin
      tmp := ((n - i + 1) * (m - i + 1)) mod MODUO;
      sol := (sol + i * tmp) mod MODUO;
  end;

  writeln(sol);

end.
```

zadatak: Faktorisanje

Dato je n brojeva. Potrebno je faktorisati svaki broj, tj. napisati ga kao proizvod prostih činioca. Svaki broj faktorisati u formatu p_1^a_1 * p_2^a_2 * ... * p_k^a_k , gde su $p_1 \leq p_2 \leq ... \leq p_k$ svi prosti činioci datog broja (u rastućem redosledu), a $a_1, a_2, ..., a_k$ - njihovi odgovarajući izložioci. Izmedju brojeva i simbola '*' i '^' ne sme biti razmaka.

Ulaz.

(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom redu standradnog ulaza nalazi se prirodan broj $n \leq 200.000$. U sledećih n redova se nalazi po jedan ceo broj b_i koga treba faktorisati ($2 \leq b_i \leq 200.000$).

Izlaz.

(Izlazne podatke ispisati na standardan izlaz.) Na standardni izlaz za svaki broj ispisati u posebnom redu njegovu faktorizaciju u gore opisanom formatu, u redosledu datim na ulazu.

Ograničenja.

U 40% test primera $n \leq 1.000$

Primer 1.

standardni ulaz	standardni izlaz
3	2^1*5^1
10	23^1
23	2^2*3^2*5^1
180	

fajl: faktorisanje.cpp

```
#include <cstdlib>
#include <cstdio>

const int MaxN = 200010;
const int MaxNum = 200000;

int n, num, p, exp, a[MaxN], leastDiv[MaxNum + 10];
bool prime[MaxNum + 10];

void eratosten(int N, bool prime[], int leastDiv[]) {

  for (int i = 0; i <= N; i++)
      prime[i] = true;

  for (int i = 2; i <= N; i++) {
    if (prime[i]) {
      leastDiv[i] = i;
      for (int j = 2; i * j <= N; j++)
        if (prime[i * j]) {
          prime[i * j] = false;
          leastDiv[i * j] = i;
```

```
        }
    }
  }
}

int main() {

  scanf("%d", &n);
  for (int i = 0; i < n; i++)
    scanf("%d", &a[i]);

  eratosten(MaxNum, prime, leastDiv);

  for (int i = 0; i < n; i++) {
    num = a[i];

    while (num > 1) {
      p = leastDiv[num];
      exp = 0;
      while (num % p == 0) {
        exp++;
        num /= p;
      }
      if (num > 1) printf("%d^%d*", p, exp);
      else printf("%d^%d\n", p, exp);
    }
  }

  return 0;
}
```

fajl: faktorisanje.pas

```
const
  MaxN = 200010;
  MaxNum = 200000;

var
  n, num, p, exp, i : longint;
  a : array[0..MaxN] of longint;
  leastDiv : array[0..MaxNum + 10] of longint;
  prime : array[0..MaxNum + 10] of boolean;


procedure eratosten(N : longint);
var
  i, j : longint;

begin

  for i := 0 to N do
      prime[i] := true;

  for i := 2 to N do
      if (prime[i]) then begin
          leastDiv[i] := i;
          for j := 2 to N div i do
              if (prime[i * j]) then begin
                  prime[i * j] := false;
                  leastDiv[i * j] := i;
              end;
      end;

end;
```

```
begin

  readln(n);
  for i := 1 to n do
      readln(a[i]);

  eratosten(MaxNum);

  for i := 1 to n do begin

      num := a[i];
      while (num > 1) do begin
          p := leastDiv[num];
          exp := 0;
          while (num mod p = 0) do begin
              exp := exp + 1;
              num := num div p;
          end;

          if (num > 1) then write(p, '^', exp, '*')
          else writeln(p, '^', exp);
      end;

  end;

end.
```

zadatak: O-Oil Fields

Sledećeg meseca održaće se licitacija za kupovinu zemlje u nedavno otkrivenim naftnim poljima. Polja su rasprostranjena ispod pravougaone doline dimenzija *X* x *Y* (*Y* predstavlja broj redova, a *X* broj kolona). Donji levi hektar označen je sa (1, 1). Hektar je jedinica mere za površinu.
Tvoja kompanija ima ekskluzivno pravo da bira zemljište pre licitacije ali i ograničen budžet. Uz pomoć satelitskih snimaka koju si dobio od Tajne Komisije na zajam, a samim tim i saznanje o rasporedu naftnih polja, neće ti biti teško da izabereš parcelu koja će tvojoj kompaniji doneti najveće zalihe nafte.

- Nafne podzemne rezerve unutar istog bazena su povezane. Dva hektara su susedna ako imaju zajedničku stranu.
- Zemlju je moguće kupiti samo u pravougaonim parcelama celobrojih mera.
- Minimalna površina koja se može kupiti je 2 hektara.
- Nafta se može crpiti sa svakog hektara koji je kupljen, kopanjem bunara samo ravno nadole.
- Cena jednog hektara je 1,000,000 dinara.
- Naftna polja ispod površine se neće preklapati. (gledano odgore)

Pošto tvoja kompanija ima dugu tradiciju u eksploataciji nafte a želi da nastavi u istom trendu tvoj zadatak je da obezbediš što je moguće više nafte dostupne sa kupljene zemlje, sa budžetom koji imaš na raspolaganju. Ako postoji više rešenja koja daju istu dostupnu površinu podzemnih naftnih polja, tada biraš

onu koja će biti najjeftinija. Postojaće samo jedno takvo rešenje.

Ulaz.
(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom standardnog redu ulaza su dva cela broja *Y* i *X*, (0 < *Y*, *X* ≤ 50) dimenzije doline. U sledećih *Y* redova su *X* celih brojeva odvojenih praznim mestima ($ID_i \le X \times Y$), sa oznakama određenog podzemnog naftnog polja. ID_i. = 0 znači da nema nafte ispod tog hektara. U sledećiem (*Y* + 2) redu je ceo broj *M*, (2,000,000 ≤ M ≤ 1,000,000,000), tvoj budžet za ovu kupovinu.
Izlaz.
(Izlazne podatke ispisati na standardan izlaz.) U prvi red standardnog izlaza ispiši 4 cela broja odvojena praznim mestima, (X_{dl} Y_{dl} X_{gd} Y_{gd}), koji predstavljaju koordinate donjeg levog i gornjeg desnog ugla parčeta doline koji ćeš kupiti sa dozvoljenim budžetom. Ako ima više rešenja sa istom količinom nafte, izaberi ono koje će ti uštedeti najviše novca. U drugi red izlaza ispiši količinu uštođenog novca. U treći red izlaza ispiši količinu nafte u hektarima koja će tvojoj kompaniji biti na raspolaganju sa ovako kupljene zemlje.
Primer 1.

standardni ulaz	**standardni izlaz**
4 4	2 2 3 3
1 1 2 2	12345
1 1 2 2	16
3 3 4 4	
3 3 4 4	
4012345	

Objašnjenje.
Sa 4,000,000 dinara budžeta, možeš da kupiš najviše 4 hektara doline. Najpovoljnije je da uzmeš 4 centralna hektara jer si time omogućio kompaniji da pristupi svim naftnim poljima koja su ispod doline. (to su polja sa oznakama *ID* = 1, 2, 3, 4). Uštođena suma je 12,345 dinara.
Primer 2.

standardni ulaz	**standardni izlaz**
4 4	2 2 2 4
0 7 7 7	1012345
3 3 5 5	14
3 5 5 5	
3 3 3 9	
4012345	

Objašnjenje.
Pravougaona površina (2,3) (3,4) omogućuje pristup do 14 hektara podzemnih rezervi doline. Do isto toliko se može dopreti kupovinom parčeta (2,2) (2,4). Medjutim pošto je veća ušteda u drugom slučaju, (2,2) (2,4) je traženo rešenje (polja sa oznakama *ID* = 3, 5, 7) .

fajl: o-oilfields.pas

```
{ Vanja Petrovic Tankovic i Nenad Bozidarevic - RAF, Beograd }

var Dolina           : array [0..51,0..51]   of LongInt;
    Dostupno         : array [0..2501]       of LongInt;
    Memo             : array [0..2501]       of LongInt;
```

```
    maxVrednost, minPolja, dlx, dly, gdx, gdy : LongInt;
    n, m, i, j, h, w, l                       : LongInt;
    pare, maxPolja, vrednost, iteracija       : LongInt;

Begin
  Readln(n, m);

  for i := n downto 1 do
  Begin
    for j := 1 to m do Read(Dolina[i,j]);
    Readln;
  end;
  Readln(pare);

  maxPolja := pare div 1000000;
  minPolja := 2501;

  for i := 1 to n do
    for j := 1 to m do
      if Dolina[i,j] > 0
        then inc(Dostupno[Dolina[i,j]]);

  for i := n downto 1 do
    for j := m downto 1 do
    Begin
      w := 0;
      while (j + w <= m) and (w+1 <= maxPolja) do
      Begin
        vrednost  := 0;
        h         := 0;
        Inc(iteracija);
        While (i + h <= n) and ((h+1)*(w+1) <= maxPolja) do
        Begin
    for l := 0 to w do
          if Memo[Dolina[i+h, j+l]] <> iteracija
            then Begin
                   Memo[Dolina[i+h, j+l]] := iteracija;
                   vrednost := vrednost + Dostupno[Dolina[i+h, j+l]];
                 end;
        if (vrednost > maxVrednost) or
           (vrednost = maxVrednost) and ((h+1)*(w+1) < minPolja)
          then Begin
                 minPolja   := (h+1)*(w+1);
                 maxVrednost:= vrednost;
                 dlx        := j;
                 dly        := i;
                 gdx        := j+w;
                 gdy        := i+h;
     end;
          inc(h);
        end;
        inc(w);
      end;
    end;
  Writeln(dlx, ' ', dly, ' ', gdx, ' ',gdy);
  Writeln(pare - minPolja * 1000000);
  Writeln(maxVrednost);
end.
```