Zadaci sa rešenjima Državno takmičenje 2007

zadatak: Plivanje

U školi je organizovan turnir u plivanju. Mali Kosta se prijavio sa namerom da pobedi u disciplini kraul. Turnir je bio organizovan u okviru nekoliko trka i posle svake trke znamo ko je bio prvi, drugi i treći. Na kraju se samo ovi rezultati uzimaju u obzir pri određivanju šampiona, tako što prvo mesto nosi *A* bodova, drugo mesto nosi *B*, dok treće mesto nosi *C* bodova. Sumiraju se bodovi za svakog takmičara i takmičar koji ima najviše bodova je šampion. Kako bodove sabira Kostin najbolji drugar iz klupe, on želi da odredi takav način bodovanja - da Kosta bude šampion ove godine.
Na takmičenje se prijavilo *n* učenika, pri čemu Kosta nosi kapicu sa rednim brojem jedan. Turnir se odvija u okviru m trka i za svaku od trka se znaju trojica prvoplasiranih. Maksimalan broj bodova za prvo mesto je p, dok je minimalan broj bodova za treće mesto 1 ($p \geq A > B > C \geq 1$). Broj bodova za drugo mesto mora biti strogo manji od broja bodova za prvo mesto i takođe broj bodova za treće mesto mora biti strogo manji od broja bodova za drugo mesto. Ukoliko je moguće, odrediti prirodne brojeve *A*, *B* i *C*, tako da je broj bodova koje je Kosta osvojio veći ili jednak broju bodova bilo kog drugog takmičara.
Ulaz:
(Ulazni podaci se nalaze u datoteci **plivanje.in**) U prvom redu ulazne datoteke se nalaze tri prirodna broja: *n*, *m* i *p*. Broj *n* predstavlja broj takmičara (plivača), a *m* je ukupan broj trka. Broj *p* je maksimalan broj bodova koje neki takmičar može da osvoji u jednoj trci. U svakom od sledećih m redova data su tri različita prirodna broja između 1 i *n*, koji predstavljaju redne brojeva takmičara koji su trku završili na prvom, drugom i trećem mestu.
Izlaz:
(Izlazne podatke upisati u datoteku **vlada.out**) U prvom i jedinom redu štampati brojeve *A*, *B* i *C*, razdvojene razmakom. U slučaju da postoji više rešenja, štampati bilo koje od njih. Ukoliko rešenje ne postoji, štampati -1.
Ograničenja:

- $3 \leq n \leq 1000$
- $1 \leq m \leq 20000$
- $1 \leq p \leq 50$
- vremensko ograničenje za izvršavanje programa je 1 s.

Primer 1:

plivanje.in	**plivanje.out**
3 5 10 1 3 2 3 1 2 1 2 3 3 1 2 3 1 2	10 8 2

Objašnjenje:
Kosta je bio dva puta prvi i tri puta drugoplasirani na turniru. Ako se za prvo mesto dobija 10 bodova, za drugo 8, a za treće 2 boda, tada će takmičari imati redom 20 + 24 + 0 = 44, 0 + 8 + 8 = 16 i 30 + 8 + 2 = 40 bodova, pa je šampion Kosta sa rednim brojem jedan.
Primer 2:

plivanje.in	**plivanje.out**
5 7 5 1 4 2 2 3 4 5 3 2 3 2 1 4 1 2 1 3 2 2 1 4	5 4 1

Primer 3:

plivanje.in	**plivanje.out**
8 4 4 3 8 1	-1

```
2 7 3
2 1 6
5 4 1
```

Objašnjenje:
lj Ako se za prvo mesto dobija 3 boda, za drugo mesto 2 i za treće 1 bod - onda će takmičar broj 2 imati 6 bodova, a Kosta 4 boda. Ako se za prvo mesto dobija 4 boda, tada će takmičar sa rednim brojem 2 imati 8 bodova, a Kosta najviše 3 + 4 = 7 bodova. Ni u jednom slučaja Kosta ne može da pobedi.

fajl: plivanje.cpp

```
/*
  DRZAVNO TAKMICENJE 2007
  ZADATAK: plivanje
  AUTOR: Aleksandar Ilic, Nis

  Svaki takmicar je oznacen rednim brojem od 1 do n. Na pocetku ucitamo sve rezultate trka
i u matrici a [n][4] pamtimo
  koliko puta je svaki takmicar bio prvi, drugi i treci. Ideja je da prodjemo po svim
mogucim brojem poena za
  prvo i drugo mesto (dva ciklusa od 1 do p), a zatim da odredimo interval u kome mora da
se nadje broj bodova
  za trece mesto - tako da takmicar sa rednim brojem 1 ima najvise poena. Na pocetku je taj
interval [1, second - 1].
  Za svako 2 <= i <= n mora da vazi
  first * a [i][1] + second * a [i][2] + third * a [i][3] <= first * a [1][1] + second * a
[1][2] + third * a [1][3]

  Za ovakvu nejednakost po nepoznatoj third dobijamo gornje ili donje ogranicenje. Na kraju
proverimo da li se
  u intervalu [minThird, maxThird] nalazi neki prirodan broj. Ukoliko je odgovor pozitivan
prekidamo i stampamo rezultat,
  a ukoliko takav broj ne postoji nastavljamo pretragu dalje.

  Vremenska slozenost algoritma je O (m + p^2 * n), dok je memorijska slozenost O (n).
  n - broj takmicara
  m - broj trka
  p - maksimalan broj bodova
  first, second, third - brojevi bodova za prvo, drugo i trece mesto
  sol - da li postoji resenje
*/

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

int n, m, p, first, second, third;
int a [1001][4];
bool sol;
double minThird, maxThird;

int main ()
{
  /* Ucitavanje podataka */
  ifstream in ("plivanje.in");
  in >> n >> m >> p;
  for (int i = 1; i <= n; i++)
      for (int j = 1; j <= 3; j++)
          a [i][j] = 0;
  for (int i = 0; i < m; i++)
  {
        in >> first >> second >> third;
        a [first][1]++;
        a [second][2]++;
        a [third][3]++;
    }
```

```
  in.close();

    /* Fiksiranje brojeva first i second i nalazenje intervala za third */
  sol = false;
    for (first = p; first >= 3; first--)
  {
        for (second = first - 1; second >= 2; second--)
        {
            minThird = 1.0;
            maxThird = second - 1;
            for (int i = 2; i <= n; i++)
            {
                double x = first * (a [i][1] - a [1][1]) + second * (a [i][2] - a [1][2]);
                double y = a [1][3] - a [i][3];
        if ((y == 0) && (x > 0))
        {
          minThird = p;
          break;
        }
                if ((y > 0) && (minThird < x / y))
          minThird = x / y;
                if ((y < 0) && (maxThird > x / y))
          maxThird = x / y;
            }
            third = (int)maxThird;
            if (third >= minThird)
            {
                sol = true;
                break;
            }
        }
    if (sol == true)
      break;
  }

  /* Stampa rezultata */
  ofstream out ("plivanje.out");
  if (sol == false)
       out << "-1" << endl;
    else
         out << first << " " << second << " " << third << endl;
   out.close();
    return 0;
}
```

zadatak: Vikendica

Dragančetu je dosadilo da živi u gradu pa je rešio da sagradi sebi vikendicu, u kojoj će u miru i tišini moći da sprovodi svoje programerske ideje. Kako je Draganče veoma vredan, rešio je da sam napravi nacrte njegove kuće iz snova. No, na početku je trebalo da odredi mesto na kome će se graditi. Kako Draganče nije baš bogat (matematika i programiranje nije bilo isplativo kako je on mislio), odredio je maksimalni budžet koji može da potroši za kupovinu placeva. Naravno, želi da vikendica bude što lepša, pa je odredio i donju granicu budžeta. Sad je u nedoumici gde da je sagradi. Pomozite Dragančetu da odredi broj mogućih mesta za gradnju vikendice.
Mesta na kojima Draganče može da gradi vikendicu data su u obliku pravougane matrice, gde polja predstavljaju placeve. Za svaki plac je data njegova cena. Vikendica je u obliku pravougaonika. Na pravougaoniku se može sagraditi vikendica ukoliko suma njenih parcela upada u granice budžeta.
Ulaz:
(Ulazni podaci se nalaze u datoteci **vikendica.in**) U prvom redu se nalaze četiri broja: *n*, *m*, *A* i *B*, koji predstavljaju broj vrsta i kolona matrice, donju granicu i gornju granicu budžeta. U narednih *n* redova nalazi se po m nenegativnih celih brojeva koji predstavljaju cene placeva.
Izlaz:
(Izlazne podatke upisati u datoteku **vikendica.out**) U prvom i jedinom redu štampati broj mogućih pozicija vikendice.

Ograničenja:

- $1 \le n, m \le 150$
- $0 \le A \le B \le 10^9$
- cene placeva su u opsegu $[0, 10^4]$
- broj rešenja ne prelazi 2^{30}
- ukupna suma cena svih placeva je manja od 2^{30}
- vremensko ograničenje za izvršavanje programa je 1 s.

Primer 1:

vikendica.in	**vikendica.out**
3 3 2 3 1 0 0 0 1 0 0 0 1	7

Objašnjenje:
Moguće pozicije za vikendicu (predstavljene sa 'o') su sledeće:

```
oox ooo oox xxx xoo xxx ooo
oox ooo oox xoo xoo ooo ooo
xxx xxx oox xoo xoo ooo ooo
```

Suma elemenata u označenim pravougaonicima je u opsegu [2,3].

Primer 2:

vikendica.in	**vikendica.out**
3 4 7 10 1 3 4 2 3 4 5 2 1 3 4 1	16

fajl: vikendica.cpp

```
/*
  DRZAVNO TAKMICENJE 2007
  ZADATAK: vikendica
  AUTOR: Andreja Ilic, Nis

    Svaki pristup zadatku vodi do problema kako da se izracuna zbir cena u
  odredjenom pravougaoniku. Zato cemo umesto samih cena placeva u matrici d pamtiti
  sume odredjenih pravougaonika, tacnije element d [i][j] ce nam predstavljati
  sumu cena u pravougaoniku sa temenima: (1,1), (i,1), (j,1), (i,j). Ovu sumu
  mozemo racunati preko formule

         d [i][j] = d [i - 1][j] + d [i][j - 1] - d [i - 1][j - 1] + cena [i][j];

    Sada mozemo linearno racunati cenu bilo kog pravougaonika. U kodu, funkcija
  Sum (x1, y1, x2, y2) nam vraca zbir cune u pravougaoniku cije je gornje levo polje
  (x1, y1) a donje desno (x2, y2).

      Vratimo se glavnom problemu. Njemu pristupamo na sledeci nacin: za svako polje
  (i, j) racunamo koliko pravougaonika postoje koji zadovoljavaju trazeni uslov,
  tako da je polje (i, j) bas donje desno polje. Naravno isporbacemo sve moguce sirine
  pravougaonika ali za njenu duzinu necemo isporabavati sve mogucnosti. Naime, kada trazimo
  broj resenja sa sirinu k mi znamo da je njena duzina sigurno manja od duzine za
  sirinu k - 1 (jer su cene nenegativne, pa je samim tim i suma rastuca) tako da cemo
  samo te mogucnosti isprobavati.

    Dakle, napravimo funkciju koja ce nam vracati broj prvougaonika koji imaju cenu
  manju ili jednaku od date vrednosti. Kao rezultat vratimo razliku tih funkvija za
  vrednosti B i A - 1.

      Vremenska slozenost algoritma je O (n^2 +  n^3), dok je memorijska slozenost O (n^2).
*/
```

```
#include <stdio.h>
#define MaxN 205

int n, m, A, B, d [MaxN][MaxN];
FILE *in, *out;

  int Sum (int x1, int y1, int x2, int y2)
  {
    return (d [x2][y2] - d [x1 - 1][y2] - d [x2][y1 - 1] + d [x1 - 1][y1 - 1]);
  }

  int Count (int Max)
  {
    int sol = 0, i, j, k, l;

    for (i = 1; i <= n; i++)
      for (j = 1; j <= m; j++)
      {
        k = 1;
        for (l = i; l >= 1; l--)
        {
          while ((k <= j) && (Sum (l, k, i, j) > Max))
            k++;
          if (k > j) break;
          sol = sol + (j - k + 1);
        }
      }
    return sol;
  }

  int main ()
  {
    int x, i, j;

    in = fopen ("vikendica.in", "r");
    out = fopen ("vikendica.out", "w");
    fscanf (in, "%d %d %d %d", &n, &m, &A, &B);

    for (i = 1; i <= n; i++)
      for (j = 1; j <= m; j++)
      {
        fscanf (in, "%d", &x);
        d [i][j] = d [i - 1][j] + d [i][j - 1] - d [i - 1][j - 1] + x;
      }
    fprintf (out, "%d\n", Count (B) - Count (A - 1));

    return 0;
  }
```

zadatak: Kredit

Profesor Đurić: *Haloooo!*
Draganče: *Dobar dan, profesore, Draganče je ovde.*
Profesor Đurić: *šta raaadiš, momčino?*
Draganče: *Upravo šetam parkom, i kad sam stigao do fontane, dobio sam ideju za onaj zadatak o kome smo juče diskutovali...*
Profesor Đurić: *A jeeee li.*
Tako se nastavio telefonski razgovor, koji je trajao *K* minuta, i ko zna koliko bi još trajao da Dragančetu nije nestalo kredita. Draganče je odmah krenuo da kupi dodatni kredit, pošto zna lokacije svih trafika u parku, i krenuo je u onu do koje mu treba najmanje vremena. čim stigne i uplati kredit, pozvaće profesora ponovo. Profesor Đurić želi da kvalitetno organizuje svoje vreme i potrebna mu je procena koliko Dragančetu treba vremena da ponovo pozove.
Ulaz:

(Ulazni podaci se nalaze u datoteci **kredit.in**) I Đurić i Draganče tačno znaju mapu parka, koja je data u obliku pravougaone matrice dimenzija *N* x *M*, gde su *N* i *M* brojevi zadati u prvom redu datoteke. U drugom redu je broj *K*. U svakom od narednih *N* redova se nalazi po *M* znakova, koji označavaju polja na mapi. Dozvoljeni znakovi su '`F`', '`x`', '`.`' i '`T`'. Znak '`F`' označava fontanu. Sve prepreke (npr. jezero, stadion, cveće...) označene su sa '`x`', dok su sa '`.`' označeni delovi parka slobodni za šetnju. Sa '`T`' su takođe označeni delovi parka gde se može proći, i u njima se nalazi trafika gde se može uplatiti kredit. Draganče se na početku razgovora nalazi kod fontane. U jednom minutu on može preći u neko od susednih polja (istok, zapad, sever ili jug), a može ostati i tu gde jeste. U toku razgovora on se šeta nasumično. Kada bude krenuo do trafike, neće praviti pauze, već ići najkraćim putem dok ne stigne do nje.

Izlaz:

(Izlazne podatke upisati u datoteku **kredit.out**) Profesor Đurić ne zna gde se Draganče šetao u toku razgovora i želi da proceni minimalno i maksimalno vreme koje je potrebno Dragančetu da dođe do trafike sa kreditom. U jedinom redu izlazne datoteke treba ispisati dva broja, *MIN* i *MAX*, odvojena razmakom; oni označavaju minimalni i maksimalni broj minuta potreban da Draganče stigne do neke od trafika.

Ograničenja:

- brojevi *N* i *M* nisu veći od 200
- *K* nije veće od 500
- broj trafika nije veći od 1000
- znak '`F`' se nalazi tačno jednom u ulaznoj datoteci
- Park je takav da Draganče uvek može stići do neke od trafika. U delu sa fontanom se ne nalazi trafika.
- vremensko ograničenje za izvršavanje programa je 1 s.

Napomena:

Ako je barem jedan od brojeva *MIN* ili *MAX* tačan, dobićete 50% poena za taj primer.

Primer 1:

kredit.in	**kredit.out**
5 5 3 x..xF .x... T..x. x.T.. ...T.	2 5

Objašnjenje:

Posle 3 minuta razgovora, Draganče će moći da se približi nekoj od trafika i za samo 2 minuta stigne do nje, a moguće je i da ostane sve vreme kraj fontane, odakle će mu trebati 5 minuta do najbliže trafike.

fajl: kredit.cpp

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

using namespace std;

FILE* fin, *fout;

int n, m, k;
char a[200][200];
int startx, starty, n_trafika;

void input() {
     char c;
     fin = fopen("kredit.in", "r");
     fscanf(fin, "%d%d%d%c", &n, &m, &k, &c);

     for (int i = 0; i < n; i++)
          fscanf(fin, "%s", &a[i]);
     fclose(fin);
```

```
}

void init() {
     input();
     for (int i = 0; i < n; i++)
         for (int j = 0; j < m; j++) {
             if (a[i][j] == 'F') {
                startx = i; starty = j;
             }
         }
}

bool in(int x, int y) {
     return (x >= 0 && x < n && y >= 0 && y < m && a[x][y] != 'X');
}

void bfs(vector<pair<int, int> > start, int dist[][200], int limit) {
     vector<pair<int, int> > next, now(start);
     for (int i = 0; i < now.size(); i++)
         dist[now[i].first][now[i].second] = 0;
     for (int i = 0; i < limit && !now.empty(); i++) {
         for (int j = 0; j < now.size(); j++) {
             pair<int, int> pos = now[j];
             int x = pos.first, y = pos.second;
             if (in(x+1, y))
                if (dist[x+1][y]==-1) {
                   next.push_back(make_pair(x+1,y));
                   dist[x+1][y] = i+1;
                }
             if (in(x-1, y))
                if (dist[x-1][y]==-1) {
                   next.push_back(make_pair(x-1,y));
                   dist[x-1][y] = i+1;
                }
             if (in(x, y+1))
                if (dist[x][y+1]==-1) {
                   next.push_back(make_pair(x,y+1));
                   dist[x][y+1] = i+1;
                }
             if (in(x, y-1))
                if (dist[x][y-1]==-1) {
                   next.push_back(make_pair(x,y-1));
                   dist[x][y-1] = i+1;
                }
         }
         now.clear();
         next.swap(now);
     }
}

void output(int min, int max) {
     fout = fopen("kredit.out", "w");
     fprintf(fout, "%d %d\n", min, max);
     fclose(fout);
}

int main(int argc, char *argv[])
{
    init();
    int dist1[n][200], dist2[n][200];

    vector<pair<int, int> > s;
    s.push_back(make_pair(startx, starty));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            dist1[i][j] = -1;
    bfs(s, dist1, k);
```

```
    vector<pair<int, int> > t;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++) {
            dist2[i][j] = -1;
            if (a[i][j] == 'T')
               t.push_back(make_pair(i, j));
        }
    bfs(t, dist2, 1000000);

    int max = 0, min = 1000000;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (dist1[i][j] != -1) {
               if (dist2[i][j] > max)
                  max = dist2[i][j];
               if (dist2[i][j] < min)
                  min = dist2[i][j];
            }
    output(min, max);
    return 0;
}
```

zadatak: LAN party

Svi ljubitelji kompjuterskih igrica znaju kakvo je uživanje pobediti protivnika sa što većom razlikom. U tom pogledu mali Kosta je veliki hedonista.
On i njegovi drugari vole da se povremeno okupljaju i igraju igrice po ceo dan. Oni se podele u dva tima i igraju *n* partija, pri čemu se nijedna partija ne završava nerešeno. Kako bi njegov tim što bolje prošao u ukupnom ishodu, Kosta vešto smišlja razne načine računanja konačnog rezultata. Omiljena strategija mu je sledeća: on *n* partija koje su odigrali podeli u najviše *k* intervala, tako da svaki interval čine uzastopne partije i svaka partija pripada tačno jednom intervalu. Za svaki interval potom računa pobednika tako što gleda čiji tim je pobedio više puta u partijama tog intervala (ako su timovi imali isto pobeda onda interval nema pobednika). Konačan rezultat Kosta onda iskazuje pobedama u intervalima, a ne u partijama.
Vaš zadatak je da pomognete Kosti da napravi takvu podelu partija na intervale da njegov tim napravi što veću razliku (po njegovom računanju), pri čemu mu je protivnički tim zadao ograničenje da ne bude više od *k* intervala.
Ako mu pomognete, Kosta će vas nagraditi besplatnom ulaznicom za sve utakmice američkog fudbala koje se održavaju na pomoćnom terenu iza stadiona.

Ulaz:
(Ulazni podaci se nalaze u datoteci **lanparty.in**) U prvom redu ulazne datoteke nalaze se prirodni brojevi *n* i *k*, razdvojeni razmakom. U narednom redu se nalazi niz od *n* slova pri čemu je svako od slova 'W' ili 'L'. Između tih slova nema razmaka. Slovo na i-toj poziciji označava ishod i-te partije: 'W' ako je pobedio Kostin tim, a 'L' ako je pobedio protivnički tim.

Izlaz:
(Izlazne podatke upisati u datoteku **lanparty.out**) Na izlaz zapišite maksimalnu razliku koju Kostin tim može ostvariti podelom partija na intervale.

Ograničenja:

- $1 \leq n \leq 100$
- $1 \leq k \leq n$
- vremensko ograničenje za izvršavanje programa je 1 s.

Primer 1:

lanparty.in	**lanparty.out**
11 5 LLWWLWLWLWW	2

Objašnjenje:
Jedan od mogućih načina da se ostvari razlika 2 je da se formira sledećih 5 intervala:

- interval 1 čine partije 1 i 2 - pobednik u ovom intervalu je protivnički tim
- interval 2 čine partije 3, 4 i 5 - pobednik u ovom intervalu je Kostin tim
- interval 3 čini samo partija 6 - pobednik je opet Kostin tim
- interval 4 čine partije 7 i 8 - oba tima imaju jednako pobeda u partijama, pa ovaj interval nema pobednika
- interval 5 čine partije 9, 10 i 11 - pobednik i u ovom intervalu je Kostin tim

Primer 2:

lanparty.in	**lanparty.out**
13 5 LLLWLLLWLLWLL	-1

Objašnjenje:
Vodite računa da najveća razlika može da bude i negativna.

fajl: lanparty.pas

```
const
  fin = 'lanparty.in';
  fout = 'lanparty.out';
  maxN = 100;

var
  n, k : Integer;
  res : Integer;

  (* win[i] = Da li je Kostin tim pobedio i-tu partiju *)
  win : array[1..maxN] of Boolean;

  (*
    s[m, j] = Maksimalna razlika koja se moze ostvariti ako se prvih m
    partija podele u tacno j intervala, pri cemu intervali mogu biti i
    prazni.
  *)
  s : array[0..maxN, 0..maxN] of Integer;

  (*
    w[i, r] = Rezultat intervala koji pocinje i-tom partijom i sadrzi r
    uzastopnih partija. 1 ako pobedjuje Kostin tim, -1 ako pobedjuje
    protivnicki tim, 0 ako je nereseno.
  *)
  w : array[1..maxN, 0..maxN] of Integer;



  procedure ReadInput();
  var
    f : Text;
    i : Integer;
    ch : Char;
  begin
    Assign(f, fin);
    Reset(f);

    Readln(f, n, k);
    for i := 1 to n do
    begin
      Read(f, ch);
      win[i] := ch = 'W';
    end;

    Close(f);
  end;
```

```
  procedure WriteOutput();
  var
    f : Text;
  begin
    Assign(f, fout);
    Rewrite(f);
    Writeln(f, res);
    Close(f);
  end;


  procedure Solve();
  var
    m, i, j, l, r, t : Integer;
  begin
    for i := 1 to n do
      for r := 0 to n-i+1 do
      begin
        w[i, r] := 0;
        for l := i to i+r-1 do      (* Najpre u w[i, r] pamtimo razliku u tom intervalu. *)
          if win[l] then
            inc(w[i, r])
          else
            dec(w[i, r]);

        if w[i, r] > 0 then w[i, r] :=  1;   (* Potom korigujemo tako da w[i, r] moze da *)
        if w[i, r] < 0 then w[i, r] := -1;   (* bude samo neka od vrednosti 1, 0 ili -1. *)
      end;


    for l := 0 to k do
      s[0, l] := 0;      (* Ako nije odigrana ni jedna partija razlika u intervalima je 0.
*)

    for m := 1 to n do
      s[m, 0] := -10000; (* Nije moguce podeliti vise od 0 partija na 0 intervala. *)

    for j := 1 to k do
      for m := 1 to n do
      begin
        s[m, j] := -1;     (* Najgora razlika u intervalima ne moze da bude manja od -1. *)
        for r := 0 to m do (* Poslednji interval sadrzi r partija. *)
        begin
          t := s[m-r, j-1] + w[m-r+1, r];
          if t > s[m, j] then
            s[m, j] := t;
        end;
      end;


    res := s[n, k];
  end;


begin
  ReadInput();
  Solve();
  WriteOutput();
end.
```

zadatak: Podniz

Kada ste uspešno pomogli Dragančetu da brzo reši zadatak Gaus, profesor Đurić se stvarno naljutio i rešio da za sledeći čas smisli stvarno težak zadatak. Profesor Đurić je sav srećan došao sutradan u školu, i dao

đacima sledeći zadatak. Na tabli je napisao niz brojeva, i traži da pronađu najkraći niz čiji su elementi brojeva iz skupa {1, 2, ..., *n*-1, *n*} (pri čemu se u nizu mogu ponavljati isti brojevi) koji nije podniz niza napisanog na tabli. Sva deca su se umirila i profesor Đurić je sedeo zadovoljno, očekujući da je miran bar za sledećih nekoliko godina. Ali previdео je dve male stvari: da je mali Draganče pametniji od ostale dece, a i da se druži sa vama, dobrim programerima. Draganče misli da je smislio način da nađe dužinu traženog niza, ali ne i koji je to niz, pa vas je zato pozvao u pomoć. Prvo treba da nađete dužinu tog niza, da bi Draganče proverio svoje rešenje, a zatim ne bi bilo loše ni da nađete taj niz. Da li će profesor Đurić ponovo biti ljut, ostaje na vama da se vidi.

Ulaz:

(Ulazni podaci se nalaze u datoteci **podniz.in**) U prvom redu nalaze se dva broja, *n* i *m*, gde je *m* broj elemenata niza zapisanog na tabli. Brojevi napisani na tabli i brojevi u traženom (pod)nizu su iz skupa {1, 2, ..., *n*-1, *n*}. U sledećih *m* redova nalazi se po jedan broj i oni predstavljaju brojeve zapisane na tabli.

Izlaz:

(Izlazne podatke upisati u datoteku **podniz.out**) U prvom redu napisati *k* - broj elemenata traženog niza. U sledećih *k* redova napisati redom po jedan element traženog niza. Ako ima više nizova iste dužine, napisati onaj koji je prvi leksikografski. Niz *b* je podniz niza *a* ako postoji niz *t* tako da za svako *i* važi $b(i) = a(t(i))$, $t(i) < t(i+1)$ (dakle, niz *b* se može dobiti od niza *a* izbacivanjem nekih elemenata). Tj. od niza 1213 podnizovi su npr. 12, 23, 123, a nisu 31, 22. Za tačno izračunat broj k dobija se 40% poena na tom test primeru.

Ograničenja:

- $n, m \leq 1000000$
- vremensko ograničenje za izvršavanje programa je 1 s.

Napomena:

U jezicima C/C++ koristite biblioteku `stdio`, a ne `iostream`, jer `iostream`-u treba više od 1 sekunde da učita 1000000 brojeva.

Primer 1:

podniz.in	**podniz.out**
2 3 1 2 1	2 2 2

Primer 2:

podniz.in	**podniz.out**
2 4 1 2 2 1	3 1 1 1

Primer 3:

podniz.in	**podniz.out**
3 10 1 2 3 3 2 1 1 2 3 1	4 1 1 2 2

fajl: podniz.cpp

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

```
#include <memory.h>

const int maxn = 1100000;

int a[maxn],ozn[maxn],d[maxn];

int main()
{
    int start = clock();
    freopen("podniz.in","r",stdin);
    freopen("podniz.out","w",stdout);

    int n,m;
    scanf("%d%d",&n,&m);

    int i,j,k,t;

    for(i= 0;i<m;i++)
    {
       scanf("%d",&(a[i]));
       a[i]--;
    }
    memset(ozn,0,n*sizeof(ozn[0]));

    k = 1; t = 0;
    for(i = m-1;i>=0;i--)
    {
        d[i] = k;
        if (ozn[a[i]]==0)
        {
            ozn[a[i]] = 1;
            t++;
            if (t==n)
            {
               memset(ozn,0,n*sizeof(ozn[0]));
               t=0;
               k++;
            }
        }
    }

    int res = k;
    printf("%d\n",res);

    memset(ozn,0,n*sizeof(ozn[0]));
    for(i = 0;i<m;i++)
    {
       if (d[i]!=k)
       {
           for (j = 0;(j<n)&&(ozn[j]);j++);
           printf("%d\n",j+1);
           k--;
           if (k==0) break;
           for (;(i<m)&&(j!=a[i]);i++);
           memset(ozn,0,n*sizeof(ozn[0]));
       }
       else ozn[a[i]]=1;
    }

    if (k==1)
    {
       for (j = 0;(j<n)&&(ozn[j]);j++);
       printf("%d\n",j+1);
    }

    return 0;
}
```

fajl: podniz.pas

```
var a,ozn,d:array[0..1100000] of Longint;
    fin,fout:Text;
    n,m,i,j,k,t,res:Longint;
    ind:boolean;
begin


    Assign(fin,'podniz.in');
    reset(fin);
    Assign(fout,'podniz.out');
    rewrite(fout);

    readln(fin,n,m);



    for i:=0 to m-1 do
    begin
       readln(fin,a[i]);
       a[i]:=a[i]-1;
    end;
    fillchar(ozn, n*sizeof(a[0]), 0);


    k := 1; t := 0;
    for i := m-1 downto 0 do
    begin
        d[i] := k;
        if (ozn[a[i]]=0) then
        begin
            ozn[a[i]] := 1;
            t:=t+1;
            if (t=n) then
            begin
               fillchar(ozn, n*sizeof(a[0]), 0);
               t:=0;
               k:=k+1;
            end;
        end;
    end;

    res := k;
    writeln(fout,res);
    ind:=true;
    fillchar(ozn, n*sizeof(a[0]), 0);
    i:=0;
    while (i<m) and ind do
    begin
       if (d[i]<>k) then
       begin
            j := 0;
     while ((j<n)and(ozn[j]>0)) do
      j:=j+1;
     writeln(fout,j+1);
            k:=k-1;
            if (k=0) then
              ind :=false
            else while ((i<m)and(j<>a[i])) do i:=i+1;
            fillchar(ozn, n*sizeof(a[0]), 0);
       end
       else ozn[a[i]]:=1;
```

```
      i:=i+1;
    end;

    if (k=1) then
    begin
      j := 0;
      while ((j<n)and(ozn[j]>0)) do j:=j+1;
      writeln(fout,j+1);
    end;

    close(fin);
    close(fout);
end.
```