Zadaci sa rešenjima Državno takmičenje 2009.

zadatak: Skupovi

Dat je skup *A* koji sadrži *n* različitih prirodnih brojeva. Odrediti koliko ima podskupova skupa *A* za koje važi da je zbir najvećeg i najmanjeg elementa jednak datom broju *m*.

Ulaz.

(Ulazni podaci se nalaze u datoteci **skupovi.in**) U prvom redu se nalaze prirodni brojevi *n* i *m* ($2 \leq n \leq 100.000$, $1 \leq m \leq 1.000.000.000$). U drugom redu se nalazi *n* različitih prirodnih brojeva, koji predstavljaju skup *A*.

Izlaz.

(Izlazne podatke upisati u datoteku **skupovi.out**) U prvom i jedinom redu treba ispisati ostatak pri deljenju broja podskupova kod kojih je zbir najvećeg i najmanjeg elementa jednak *m* i broja 1.000.000.007.

Primer 1.

skupovi.in	skupovi.out
5 9 7 2 9 5 4	5

Primer 2.

skupovi.in	skupovi.out
6 11 2 4 6 8 10 12	0

Objašnjenje.

U prvom primeru traženi podskupovi su {2, 7}, {2, 4, 7}, {2, 5, 7}, {2, 4, 5, 7} i {4, 5}. Primetimo da skup {9} ne zadovoljava uslove zadatka, pošto je zbir najvećeg (broj 9) i najmanjeg (broj 9) elementa jednak 18. U drugom primeru ne postoji podskup kod koga je zbir najvećeg i najmanjeg elementa neparan broj, pa je zato rešenje 0.

fajl: skupovi.cpp

```
#include <fstream>
#include <iostream>
#include <cstdlib>

#define MAXN 100001
#define MOD 1000000007

using namespace std;

long *a, *deg;
long n, m, sol;

void input()
{
  ifstream inFile;
  inFile.open("skupovi.in");
  inFile >> n >> m;
  a = new long [n];
  for (int i = 0; i < n; i++)
    inFile >> a [i];
  inFile.close();
}

int compare (const void *a, const void *b)
{
  return *(long *)a - *(long *)b;
}

void solve()
{
  qsort(a, n, sizeof(long), compare);
  deg = new long [n + 1];
  deg [0] = 1;
  for (int i = 1; i <= n; i++)
```

```
  {
    deg [i] = deg [i - 1] * 2;
    if (deg [i] > MOD)
      deg [i] -= MOD;
  }
  sol = 0;

  int left = 0;
  int right = n - 1;
  int count = 0;
  while (right > left)
  {
    if (a [left] + a [right] == m)
    {
      sol = sol + deg [right - left - 1];
      if (sol > MOD)
        sol -= MOD;
      count++;
    }
    if (a [left] + a [right] <= m)
      left++;
    else
      right--;
  }
  if (a [left] + a [right] == m)
  {
    count++;
    sol = sol + 1;
    if (sol > MOD)
      sol -= MOD;
  }
  cout << count << endl;
}

void output()
{
  ofstream outFile;
  outFile.open("skupovi.out");
  outFile << sol << endl;
  outFile.close();
}

int main()
{
  input();
  solve();
  output();
  check();
  return 0;
}
```

zadatak: Knjižica

Kići i Ćiki su već dugo najbolji drugari. Nažalost, Kići je izgubio pamćenje i Ćiki je odlučio da mu spremi specijalno iznenađenje kako bi mu pomogao da se seti svega. On je sve njihove zajedničke uspomene zapisivao u jednu zelenu sveskicu, i sada on želi da od toga napravi jednu od onih knjižica gde je čitaocu povremeno ponuđeno da izabere kako će se priča nastaviti. Na kraju svakog dela priče, čitalac bira da li će nastaviti onako kako bi Kići nastavio ili onako kako bi Ćiki nastavio, s tim što se sa nekih delova ne može nastaviti dalje (na njima se priča završava), a na pojedinim delovima postoji samo jedna mogućnost da se nastavi priča (nekad je to Ćikijeva a nekad Kićijeva ideja). U toku jednog čitanja ne možemo dva puta da se nađemo na istom delu i do svakog dela priče se može stići na jedinstven način. Svaki deo priče se nalazi na po jednoj stranici.
Delovi ove priče su numerisani brojevima od 1 do n (gde je n broj delova priče) ali pošto je Ćiki perfekcionista on želi da ih renumeriše (opet brojevima od 1 do n). Ćiki je uvek voleo male stvari, a Kići

velike, pa zato Ćiki želi da: ako sa dela priče koji je na stranici *a* prelazimo na deo priče koji je na stranici *b* važi:

- Ako smo izabrali ono što bi Kići uradio, onda je $b > a$, i redni broj svake stranice do koje ćemo kasnije u toku čitanja priče moći da dođemo je veći od *a*
- Ako smo izabrali ono što bi Ćiki uradio, onda je $b < a$, i redni broj svake stranice do koje ćemo kasnije u toku čitanja priče moći da dođemo je manji od *a*

Pomozite Ćikiju da složi stranice kako bi što pre mogao da da knjižicu Kićiju (Ćiki je sjajan programer, ali je trenutno previše tužan da bi razmišljao o ovom problemu).

Ulaz.

(Ulazni podaci se nalaze u datoteci **knjizica.in**) U prvom redu nalaze se tri broja *n*, *k* i *c* ($1 \leq n \leq 250.000$, $1 \leq k, c \leq n$). Zatim se u narednih *k* redova nalaze po dva broja, *a* i *b*, koji označavaju da sa dela priče obeleženog brojem *a* ako poslušamo Kićija prelazimo na deo obeležen brojem *b*. Slično se u narednih *c* redova nalaze po dva broja, *a* i *b*, koji označavaju da sa dela priče obeleženog brojem *a* ako poslušamo Ćikija prelazimo na deo obeležen brojem *b*. Pre renumerisanja početak priče je na delu obeleženom brojem 1 (obratite pažnju da posle renumeracije ne mora biti).

Izlaz.

(Izlazne podatke upisati u datoteku **knjizica.out**) Treba da se sastoji od *n* redova - u *i*-tom redu nalazi se broj stranice na kojoj će se deo priče obeležen brojem *i* nalaziti posle renumeracije. Ako postoji više rešenja, štampati bilo koje.

Primer 1.

knjizica.in	knjizica.out
7 3 3	3
3 6	2
1 3	6
5 7	1
1 2	4
3 5	7
2 4	5

fajl: knjizica.cpp

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

const int maxN = 500000;

int levi[maxN + 1];
int desni[maxN + 1];
int strana[maxN + 1];
int brojac;

void numerisi(int deo) {
     if (deo == -1) return;
     numerisi(levi[deo]);
     brojac++;
     strana[deo] = brojac;
     numerisi(desni[deo]);
}

int main(){
    int n, k, c, a, b;

    freopen("knjizica.in", "r", stdin);
    scanf("%d %d %d",&n, &k, &c);
    for (int i = 1; i <= n; i++) {
        levi[i] = -1;
        desni[i] = -1;
    }
    for (int i = 1; i <= k; i++) {
```

```
        scanf("%d %d", &a, &b);
        desni[a] = b;
    }
    for (int i = 1; i <= c; i++) {
        scanf("%d %d", &a, &b);
        levi[a] = b;
    }

    brojac = 0;
    numerisi(1);

    freopen("knjizica.out", "w", stdout);
    for (int i = 1; i <= n; i++)
        printf("%d\n", strana[i]);

    return 0;
}
```

zadatak: Šifra

Draganče radi za kontraobaveštajnu službu. On je otkrio da se šifrovane poruke neprijatelja uvek dobijaju linearnim preslikavanjem. Najpre se svako slovo predstavi brojem od 0 do $n - 1$, gde je n broj slova u pismu (ove brojeve ćemo zvati indeksima slova). Potom se šifra zada u obliku $y = (a + b \cdot x) \bmod n$, gde je $nzd(b, n) = 1$. To znači da se slovo predstavljeno brojem x šifrira slovom koje je predstavljeno brojem y ($y = (a + b \cdot x) \bmod n$). Pored saznanja o načinu šifrovanja poruka, Draganče ima i podatke o učestanosti svih slova u jeziku neprijatelja. Za svako slovo x se zna očekivani procenat pojavljivanja $p(x)$ u proizvoljnom tekstu. Treba pomoći Dragančetu da dešifruje jednu izuzetno važnu poruku uz pomoć statistike i saznanja o linearnom šifrovanju.
Potrebno je izvršiti dešifrovanje tako da se statistika u dobijenom tekstu što više poklapa sa očekivanom statistikom. Kvalitet poklapanja za odredjeno slovo se meri tako što se najpre izračuna učestanost tog slova u dešifrovanom tekstu $dc(x)$ (broj pojavljivanja slova x podeljen sa ukupnim brojem slova u tekstu). Potom se kvalitet poklapanja $q(x)$ računa kao $(p(x)-dc(x))^2$. Kvalitet poklapanja čitavog dešifrovanog teksta se računa kao suma kvaliteta poklapanja svih slova.

Ulaz.

(Ulazni podaci se nalaze u datoteci **sifra.in**) U prvom redu se nalazi prirodan broj n, $3 \leq n \leq 26$, ukupan broj slova u neprijateljskom alfabetu. Sledi n redova i u svakom je dato jedno slovo iz alfabeta i jedan broj, razdvojeni razmakom. Slovo u prvom redu ima indeks 0, u drugom 1, ... u n-tom redu ima indeks $n - 1$. Slovo je uvek predstavljeno kao veliko slovo iz engleskog alfabeta, dok broj predstavlja procenat učestanosti datog slova u proizvoljnom tekstu. Procenat je zadat sa dve decimale. Slovo imaju različite učestanosti, a ukupan zbir svih učestanosti je 100. U sledećem redu je ceo broj L ($10 \leq L \leq 1000$) i on predstavlja dužinu šifrovanog teksta. U poslednjem redu, nalazi se tekst od tačno L slova (bez razmaka i znakova interpunkcije) koji predstavljaju šifrovani tekst. Sva slova koja se nalaze u šifrovanom tekstu su prethodno već zadata sa svojim učestanostima.

Izlaz.

(Izlazne podatke upisati u datoteku **sifra.out**) U jedinom redu izlaza treba ispisati dešifrovani tekst, dešifrovan preko linearnog preslikavanja, a koji najviše odgovara zadatoj statistici.

Primer 1.

sifra.in	**sifra.out**
4 B 42.25 A 20.00 Z 10.15 E 27.60 8 AZABAZEBAZ	BEBABEZABE

Napomena.

Na osnovu ulaza imamo se slova predstavljaju brojevima od 0 do 3 na sledeći nachin: $B \rightarrow 0$, $A \rightarrow 1$, $Z \rightarrow 2$ i $E \rightarrow 3$. Preslikavanje kod kog se dobija najbolje preklapanje je $y = (a + b \cdot x) \bmod 4$, gde su $a = 1$ i $b = 3$.

fajl: sifra.cpp

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

const int MAXL = 1000+ 5, MAXN = 26 + 5;
int n, l;
char c[MAXN];
int index[MAXL];
char s[MAXL];
char res[MAXL];
double exp_freq[MAXN];
double freq[MAXN];

void input() {
  ifstream fin("sifra.in");
  fin >> n;
  // ucitavamo ocekivane frekvencije
  for (int i = 0; i < n; i++) {
    fin >> c[i] >> exp_freq[i];
    freq[i] = 0;
    exp_freq[i] /= 100.;
  }
  fin >> l;
  // ucitavamo poruku i azuriramo frekvencije
  for (int i = 0; i < l; i++) {
    fin >> s[i];
    for (int j = 0; j < n; j++)
      if (s[i] == c[j]) {
        index[i] = j;
        freq[j] ++;
      }
  }
  for (int i = 0; i < n; i++)
    freq[i] /= n;
  fin.close();
}

int nzd(int a, int b) {
  if (a ==0) return b;
  if (a < b)
    return nzd(b%a, a);
  else
    return nzd(a%b, b);
}

void resi() {
  double minqual = 1e15;
  int bestA, bestB;
  for (int b = 1; b < n; b++)
    if (nzd(b, n) == 1)
      for (int a = 0; a < n; a++) {
        double qual = 0;
        for (int i = 0; i < n; i++) {
          int m = (i * b + a) % n;
          qual += (freq[i] - exp_freq[m]) * (freq[i] - exp_freq[m]);
        }
        if (qual < minqual) {
          minqual = qual;
          bestA = a;
          bestB = b;
        }
      }
  for (int i = 0; i < l; i++)
    for (int j = 0; j < n; j++)
      if (c[j] == s[i]) {
```

```
        res[i] = c[(j*bestB+bestA)%n];
      }
  res[l] = 0;

}

void output() {
  ofstream fout("sifra.out");
  fout << res << endl;
  fout.close();
}

int main() {
  input();
  resi();
  output();
  return 0;
}
```

zadatak: Klizanje

Učiteljica Danka je odlučila da svoje đake odvede na klizanje. Međutim, pošto je jedino ona bila raspoložena za takvu avanturu, i đaci ostalih razreda su odlučili da se pridruže. I tako je ona sa n učenika otišla na obližnje klizalište. Kada su došli, ona je zamolila svakog đaka da joj kaže odakle će početi da se kliza, i u kom pravcu će se klizati. Kako je Danka veoma pametna učiteljica, ona je na osnovu godina svakog đaka i njihove snage i osobina zaključila i kojom brzinom će se svako klizati. Pošto veoma dugo radi kao učiteljica, ima dovoljno iskustva da sve đake može da kontroliše čak i ako se ne pomera (tj. sve vreme stoji na samo jednom mestu). Danka slabo vidi, a bezbednost dece je na prvom mestu, pa je odlučila da ponese naočare. Ali ne bilo kakve naočare, već specijalne naočare na kojima može da se namesti koliko daleko se vidi s njima (a može videti i levo i desno od svoje pozicije onoliko daleko koliko je podesila). Da bi bila sigurna da je sve u redu, odlučila je da podesi daljinu naočara tako da bar u jednom momentu vidi bar k učenika. Pošto ste Vi njen najbolji učenik, zamolila vas je da joj izračunate kolika je najmanja moguća daljina na koju treba da podesi naočare i gde treba da stoji tako da to važi.
Klizalište je predstavljeno jednom beskonačnom pravom, pozicije učiteljice i đaka predstavljene su x-koordinatom na toj pravoj.

Ulaz.

(Ulazni podaci se nalaze u datoteci **klizanje.in**) U prvom redu datoteke se nalaze brojevi n ($4 \leq n \leq 1000$) i k ($1 \leq k \leq n$). U svakom od narednih n redova se nalaze 2 broja: prvi broj $x[i]$ ($0 \leq x[i] \leq 500.000$) označava početnu poziciju đaka, a drugi broj $v[i]$ ($-1000 \leq v[i] \leq 1000$) brzinu đaka (odnosno koliko će se pomeriti svake sekunde, ako je brzina negativna, to znači da se kreće ulevo po x osi). Dva ili više učenika mogu imati istu početnu poziciju, ali u tom slučaju ne mogu imati jednake brzine.

Napomena.

Đaci ne menjaju brzinu tokom spuštanja, i nijedan đak neće "pregaziti" učiteljicu.

Izlaz.

(Izlazne podatke upisati u datoteku **klizanje.out**) U izlaznu datoteku treba ispisati traženu najmanju moguću daljinu. Dozvoljena relativna greška je 10^{-8}. Ako je r korektno rešenje, a s rešenje koje proizvede vaš program onda treba da važi:

$2|r-s| / (|r|+|s|) \leq 10^{-8}$

Primer 1.

klizanje.in	klizanje.out
4 3 0 1 2 4 5 -2 7 0	1.5

Objašnjenje.

U trenutku 0.5, poslednja tri klizača će biti na pozicijama 4, 4, 7, redom pa će Danka, ako stoji na poziciji 5.5 i podesi daljinu na 1.5, videti njih trojicu.

fajl: klizanje.cpp

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

const int TEST = 0;

const int MAXN = 10000;
const int MAXP = MAXN * MAXN;

typedef struct {
   int x;
   int v;
} Djak;


typedef struct {
   int dx;
   int dv;
   int i;
   int j;
} Presek;

Djak djaci[MAXN];
Presek preseci[MAXP];
int poz[MAXN];
int koji[MAXN];



inline double abs (double a)
{
 return a>=0?a:-a;
}
inline double min(double a, double b) {
 return a<b?a:b;
}
inline int min(int a, int b) {
 return a<b?a:b;
}
inline int max(int a, int b) {
 return a>b?a:b;
}

int presek(Djak* a, int i, int j, Presek& p) {
     p.dx = a[i].x - a[j].x;
     p.dv = a[j].v - a[i].v;
     p.i = i;
     p.j = j;
     return p.dx * p.dv > 0;
}

int op(const void * a, const void * b) {
    Djak* pa = (Djak *)a;
    Djak* pb = (Djak *)b;
    if (pa->x != pb->x)
       return (pa->x) - (pb->x);
    return (pa->v) - (pb->v);
}

int cmpPresek(const void * a, const void * b) {
    Presek* pa = (Presek *)a;
    Presek* pb = (Presek *)b;
    int v =  pa->dx * pb->dv - pb->dx * pa->dv;
    if (v!=0)
       return v;

    int r = djaci[pa->i].x - djaci[pb->i].x;
```

```
    if (r!=0) return r;
    r = djaci[pa->j].x - djaci[pb->j].x;
    if (r!=0) return r;
    r = pa->i - pb->i;
    if (r!=0) return r;
    r = pa->j - pb->j;
    return r;
}

void razmeni(int* poz, int* koji, int i, int j)
{
     koji[poz[i]]=j;
     koji[poz[j]]=i;
     int pom = poz[i];
     poz[i] = poz[j];
     poz[j] = pom;
}

double vreme(Presek& p) {
       return 1.0*p.dx/p.dv;
}

double pozicija(Djak& a, double t) {
       return (t*a.v+a.x);
}

double rastojanje(Djak* a, int i, int j, Presek& p) {
       double t= vreme(p);
       return abs(pozicija(a[i],t)-pozicija(a[j],t));
}



double calc(int n, int k, Djak* a) {
    qsort(a, n, sizeof(Djak), op);

    int broj = 0;
    int i,j;
    double res = 1e50;
    for (i=0;i<n;i++) {
        if (i+k-1 < n)
           res = min(res, (double)abs(a[i].x - a[i+k-1].x));
        if (TEST)
           printf("%d %d\n", a[i].x, a[i].v);
        if (i+1<n)
           if (a[i].x==a[i+1].x && a[i].v==a[i+1].v)
              printf("imamo istih %d %d\n", a[i].x, a[i].v);
        poz[i] = i;
        koji[i] = i;
    }
    if (TEST)
       printf("\n");

    for (i = 0;i<n;i++)
        for(j = i+1;j<n;j++)
        {
            if (presek(a, i, j, preseci[broj])>0)
               broj++;
        }
    int start = clock();

    qsort(preseci, broj, sizeof(Presek), cmpPresek);


    printf("sortiranje %d %d\n", broj, clock()-start);
```

```
    int p;
    for (p = 0;p<broj;p++) {
        i = preseci[p].i;
        j = preseci[p].j;
        int c;
        if (TEST) {
           for (c=0;c<n;c++)
               printf("%d  ", koji[c]);
           printf("\n");
           printf("%d %d\t%d %d\n", i, j, poz[i], poz[j]);
        }

        if (koji[poz[i]]!=i || koji[poz[j]]!=j)
           printf("GREEESKA %d %d\n", poz[i], poz[j]);

        if (abs(poz[i]-poz[j])>1) {
           printf("BAAAD %d %d\n", poz[i], poz[j]);
        }
        razmeni(poz, koji, i, j);
        int left = min(poz[i], poz[j]);
        int right = left + k-1;
        double r ;
        if (right <n)
        {
           r = rastojanje(a, koji[left], koji[right], preseci[p]);
           if (TEST)
              printf("\t%d %d %lf\n", left , right, r);
           res = min(res, r);
        }
        right = max(poz[i], poz[j]);
        left = right - k+1;
        if (left>=0)
        {
           r =  rastojanje(a, koji[left], koji[right], preseci[p]);
           if (TEST)
              printf("\t%d %d %lf\n", left , right, r);
           res = min(res, r);
        }
    }
    return res;
}


int main() {
    FILE* fin = fopen("klizanje.in", "r");
    FILE* fout = fopen("klizanje.out", "w");


    int n,k;
    fscanf(fin, "%d%d", &n, &k);
    int i;
    for(i = 0;i<n;i++)
       fscanf(fin, "%d%d", &(djaci[i].x), &(djaci[i].v));

    double res = calc(n, k, djaci);

    fprintf(fout, "%lf\n", res);
    fclose(fin);
    fclose(fout);

    system("pause");
    return 0;
}
```

zadatak: Vašari

Došla je sezona vašara. Trgovac Vasa bi da ove sezone zaradi dosta para pa hoće da sve isplanira unapred. U više gradova se organizuju vašari. Vasa zna kog dana u nekom gradu počinje vašar i koliko dana traje. Isto tako zna kolika će gužva biti na tom vašaru i koliko može da zaradi u jednom danu.
Ali Vasa ne planira da prodaje samo u jednom gradu. On planira da se seli iz jednog grada u drugi i želi da mu mi odredimo koliko najviše može da zaradi.

Ulaz.

(Ulazni podaci se nalaze u datoteci **vasari.in**) U prvom redu se nalaze dva broja *n* i *p* ($1 \le p \le n \le 1000$). Broj *n* predstavlja broj gradova u kojima se održavaju vašari, a *p* je redni broj grada u kojem se Vasa nalazi prvog dana. U svakom od sledećih *n* redova (svaki red je vezan za jedan grad) se nalaze po *n* + *3* broja: *a*, *b*, *c*, *v[1], v[2], ..., v[n]* ($1 \le a, b \le 200$, $1 \le c \le 1000$, $0 \le v[i] \le 400$). Broj *a* predstavlja kog dana u tom gradu počinje vašar, *b* predstavlja koliko dana traje, *c* predstavlja koliko Vasa može da zaradi u jednom danu na tom vašaru. Broj *v[i]* predstavlja koliko je potrebno dana da Vasa pređe iz tog grada u grad *i*. Primetite da godina može imati i više od 365 dana (ali ne više od 400 dana).

Izlaz.

(Izlazne podatke upisati u datoteku **vasari.out**) Treba da se sastoji od samo jednog broja koji predstavlja koliko Vasa može da zaradi.

Primer 1.

vasari.in	**vasari.out**
3 2 5 6 5 0 3 3 1 5 2 2 0 1 7 8 3 1 2 0	37

Objašnjenje.

Prva dva dana Vasa prodaje u drugom gradu i zaradi 4. Onda dva dana ne prodaje nego se seli u prvi grad. Tamo prodaje tokom celog vašara i zaradi 30. Onda tri dana prelazi u treći grad i u njemu prodaje samo jedan dan, i zaradi 3. Ukupno je zaradio 4 + 30 + 3 = 37.

fajl: vasari.c

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

int main()
{
  long i,j,d,p;
  int n,st;
  int  poc[1005],duz[1005],z[1005];
  int *mat[1005];
  int x;
  long *res[1005];
  long max=0;
  freopen("vasari.in","r",stdin);
  freopen("vasari.out","w",stdout);
  scanf("%d%d",&n,&st);
  st--;
  for(i=0;i<n;i++)
  {
    scanf("%d%d%d",&poc[i],&duz[i],&z[i]);
    if(poc[i]+duz[i]>max)
      max=poc[i]+duz[i];
    mat[i]=(int*)malloc(1005* sizeof(int));
    res[i]=(long*)malloc(505*sizeof(long));
    for(j=0;j<n;j++)
    {
      scanf("%d",&mat[i][j]);
    }
  }
  for(i=0;i<n;i++)
  {
    res[i][0]=-1;
  }
  res[st][0]=0;
  for(d=1;d<=max;d++)
```

```
  {
    for(i=0;i<n;i++)
    {
      res[i][d]=res[i][d-1];
      if(res[i][d]>=0 && d>=poc[i] && d<poc[i]+duz[i])
  res[i][d]+=z[i];
      for(j=0;j<n;j++)
      {
  if(i==j) continue;
  p=d-mat[j][i];
  if(p<0 || res[j][p]==-1) continue;
  if(res[j][p]>res[i][d])
    res[i][d]=res[j][p];
      }
    }
  }

  d=0;
  for(i=0;i<n;i++)
  {
    if(res[i][max]>d)
      d=res[i][max];
  }
  printf("%d\n",d);

  return 0;
}
```