Zadaci sa rešenjima Savezno takmičenje 2006.

zadatak: Proizvod

Trgovac Joca je u svojoj radnji uveo sistem multiplikativnih cena. Naime, za svaki artikal postoji osnovna cena i Jocina marža, a kupac je dužan da za artikal plati iznos jednak njihovom proizvodu. Trgovac Joca trenutno vrši popis i postavlja nove cene. Za jedan artikal on ima određeni broj pločica, pri čemu je na svakoj ispisana po jedna cifra. Od tih pločica on želi da sastavi osnovnu cenu i maržu za taj artikal tako da iznos koji kupac mora da plati za njega bude maksimalan. S obzirom da je on sva računanja obavio na papiru, on vas je zamolio da vi na računaru nađete poslednje cifre tog maksimalnog iznosa, na osnovu koga će on proveriti ispravnost svog računanja. I cena i marža moraju da sadrže makar jednu pločicu.

Ulaz:

U prvom redu ulaznog tekstualnog fajla **proizvod.in** nalaze se dva prirodna broja N i M razdvojenih blanko znakom ($2 \le N \le 105$, $1 \le M \le 8$). N predstavlja broj pločica za dati artikal, a M predstavlja broj poslednjih cifara maksimalnog iznosa koje interesuju trgovca Jocu. U drugom redu nalazi se N dekadnih cifara koje predstavljaju cifre na pločicama. Pre i između cifara u drugom redu ne postoji ni jedan blanko znak.

Izlaz:

U prvi i jedini red izlaznog tekstualnog fajla **proizvod.out** upisati poslednjih M cifara maksimalnog iznosa koji se za dati artikal može dobiti pomoću zadatih pločica. Ukoliko je broj cifara maksimalnog iznosa manji od M, onda upisati ceo iznos dopunjen vodećim nulama tako da je ukupan broj cifara u ispisu jednak M. Pre i između cifara ne upisivati ni jedan blanko znak.

Primer:

proizvod.in	proizvod.out
4 2 2397	16

Objašnjenje:

Najveći iznos se dobija ako je osnovna cena 92, a Jocina marža 73 (ili obrnuto). Tada je iznos 6716, a u ispisu se prikazuju poslednje dve cifre, tj. 16.

Primer:

proizvod.in	proizvod.out
3 3 501	050

Objašnjenje:

Najveći iznos se dobija ako je osnovna cena 5, a Jocina marža 10 (ili obrnuto); ili ako je osnovna cena 50, a marža 1 (ili obrnuto). Tada je iznos 50, a u ispisu se prikazuje pun iznos sa dopisanom jednom nulom, tj. 050.

Rešenje

Problem opisan u zadatku se može iskazati kao problem sastavljanja dva broja od zadatih cifara, tako da je njihov proizvod maksimalan. Neka su to brojevi A i B i neka se oni mogu napisati kao:
$A=a_n a_{n-1}\ldots a_1 a_0$ i $B=b_m b_{m-1}\ldots b_1 b_0$
Očigledno je da će u oba broja cifre biti poređane u nerastućem redosledu, posmatrajući počev od cifre najveće težine, odnosno:
$a_n \ge a_{n-1} \ge \ldots \ge a_1 \ge a_0$ i $b_m \ge b_{m-1} \ge \ldots \ge b_1 \ge b_0$
Neka su na raspolaganju cifre (bez gubljenja opštosti možemo ih posmatrati u uređenom redosledu):
$c_t \ge c_{t-1} \ge \ldots \ge c_1 \ge c_0$ ($t=n+m+1$)
Naš zadatak možemo posmatrati kao određivanje koja cifra pripada kom od dva broja, pošto znamo kako se cifre raspoređuju unutar jednog broja. Radi lakšeg zapisa, uvedimo da oznaka $A_{i,j}$ predstavlja broj $a_i a_{i-1}\ldots a_{j+1} a_j$ i analognu oznaku $B_{i,j}$.
Očigledno, najveća data cifra mora biti cifra najveće težine jednog od ova dva broja. Neka je to broj A. Dakle, $c_t \ge a_n$. Pokušajmo da utvrdimo kom broju pripada cifra c_{t-1}. Postoje dve mogućnosti: $c_{t-1} \ge b_m$ ili $c_{t-1} \ge a_{n-1}$. Dokazaćemo da je $c_{t-1} \ge b_m$. Pretpostavimo suprotno, tj. da je $c_{t-1} \ge a_{n-1}$. Tada je proizvod brojeva A i B:
$AB^{(1)}=(c_t 10^n+c_{t-1}10^{n-1}+A_{n-2,0})\ (b_m 10^m+B_{m-1,0})=c_t b_m 10^{n+m}+c_t 10^n B_{m-1,0}+c_{t-1}b_m 10^{n+m-1}+c_{t-1}10^{n-1}B_{m-1,0}+A_{n-2,0}b_m 10^m+A_{n-2,0}B_{m-1,0}$
Ako bi cifre a_{n-1} i b_m zamenile mesta, dobili bismo proizvod:
$AB^{(2)}=(c_t 10^n+b_m 10^{n-1}+A_{n-2,0})\ (c_{t-1}10^m+B_{m-1,0})=c_t c_{t-1}10^{n+m}+c_t 10^n B_{m-1,0}+b_m c_{t-1}10^{n+m-1}+b_m 10^{n-1}B_{m-1,0}+A_{n-2,0}c_{t-1}10^m+A_{n-2,0}B_{m-1,0}$
Razliku ovih proizvoda možemo pisati kao:
$AB^{(2)}-AB^{(1)}=c_t 10^{n+m}(c_{t-1}-b_m)+10^{n-1}B_{m-1,0}(b_m-c_{t-1})+A_{n-2,0}10^m(c_{t-1}-b_m)=(c_{t-1}-b_m)\ (c_t 10^{n+m}-10^{n-1}B_{m-1,0}+10^m A_{n-2,0})$

S obzirom na to što je $c_{t-1} \geq b_m$, izraz u prvoj zagradi je nenegativan. Ako je $c_t \geq 0$, onda je i $B_{m-1,0} \geq 0$ (nijedna cifra u B nije veća od c_t), pa je i izraz u drugoj zagradi nenegativan (tačnije jednak 0). Ako je $c_t > 0$, onda je $c_t 10^{n+m} \geq 10^{n+m}$, a pošto iz $B_{m-1,0} < 10^m$ sledi $10^{n-1} B_{m-1,0} < 10^{n+m-1}$, tada je $c_t 10^{n+m} - 10^{n-1} B_{m-1,0} > 10^{n+m} - 10^{n+m-1} > 0$, pa je i u ovom slučaju izraz u drugoj zagradi nenegativan (u ovom slučaju strogo pozitivan). Time smo dokazali da uvek važi:

$AB^{(2)} - AB^{(1)} \geq 0$

Samo za $c_{t-1} = b_m$ važi znak jednakosti. Za $c_{t-1} > b_m$ važi da je $AB^{(2)} - AB^{(1)} > 0$, što je kontradikcija našem izboru lokacije za cifru c_{t-1}. Dakle, potrebno je odabrati $c_{t-1} \geq b_m$.

Pretpostavimo da smo rasporedili cifre $c_t,...,c_{k+1}$ među ciframa $a_n,...,a_{i+1},b_m,...,b_{j+1}$ ($t - k = n - i + m - j$). Za cifru c_k mora važiti $c_k \geq b_j$ ili $c_k \geq a_i$. Ukoliko je $A_{n,i+1} \geq B_{m,j+1}$, prethodni izbor je proizvoljan (zbog simetrije). U suprotnom, bez gubljenja opštosti, pretpostavimo da je $A_{n,i+1} > B_{m,j+1}$. Dokazaćemo da je tada $c_k \geq b_j$.

Pretpostavimo suprotno, tj. da je $c_k \geq a_i$. Tada je proizvod brojeva A i B:

$AB^{(1)} = (A_{n,i+1}10^{i+1} + c_k 10^i + A_{i-1,0})\ (B_{m,j+1}10^{j+1} + b_j 10^j + B_{j-1,0}) = A_{n,i+1}B_{m,j+1}10^{i+j} + 2 + A_{n,i+1}b_j 10^{i+j+1} + A_{n,i+1}B_{j-1,0}10^{i+1} + c_k B_{m,j+1}10^{i+j+1} + c_k b_j 10^{i+j} + c_k B_{j-1,0}10^i + A_{i-1,0}B_{m,j+1}10^{j+1} + A_{i-1,0}b_j 10^j + A_{i-1,0}B_{j-1,0}$

Ako bi cifre a_i i b_j zamenile mesta, dobili bismo proizvod:

$AB^{(1)} = (A_{n,i+1}10^{i+1} + b_j 10^i + A_{i-1,0})\ (B_{m,j+1}10^{j+1} + c_k 10^j + B_{j-1,0}) = A_{n,i+1}B_{m,j+1}10^{i+j} + 2 + A_{n,i+1}c_k 10^{i+j+1} + A_{n,i+1}B_{j-1,0}10^{i+1} + b_j B_{m,j+1}10^{i+j+1} + b_j c_k 10^{i+j} + b_j B_{j-1,0}10^i + A_{i-1,0}B_{m,j+1}10^{j+1} + A_{i-1,0}c_k 10^j + A_{i-1,0}B_{j-1,0}$

Razliku ovih proizvoda možemo pisati kao:

$AB^{(2)} - AB^{(1)} = A_{n,i+1}10^{i+j+1}(c_k - b_j) + B_{m,j+1}10^{i+j+1}(b_j - c_k) + B_{j-1,0}10^i(b_j - c_k) + A_{i-1,0}10^j(c_k - b_j) = (c_k - b_j)\ (A_{n,i+1}10^{i+j+1} - B_{m,j+1}10^{i+j+1} - B_{j-1,0}10^i + A_{i-1,0}10^j)$

S obzirom na to što je $c_k \geq b_j$, izraz u prvoj zagradi je nenegativan. S obzirom na to što je $A_{n,i+1} > B_{m,j+1}$, važi:

$A_{n,i+1}10^{i+j+1} - B_{m,j+1}10^{i+j+1} = (A_{n,i+1} - B_{m,j+1})10^{i+j+1} \geq 10^{i+j+1}$

S druge strane je $B_{j-1,0} < 10^j$, pa je $B_{j-1,0}10^i < 10^{i+j}$. Odatle je

$A_{n,i+1}10^{i+j+1} - B_{m,j+1}10^{i+j+1} - B_{j-1,0}10^i > 10^{i+j+1} - 10^{i+j} > 0$,

pa je izraz u drugoj zagradi pozitivan. Time smo dokazali da uvek važi:

$AB^{(2)} - AB^{(1)} \geq 0$

Samo za $c_k = b_j$ važi znak jednakosti. Za $c_k > b_j$ važi da je $AB^{(2)} - AB^{(1)} > 0$, što je kontradikcija našem izboru lokacije za cifru c_k. Dakle, potrebno je odabrati $c_k \geq b_j$.

Sada možemo sastaviti algoritam:

```
sortirati niz cifara c u opadajući poredak c[t]≥c[t-1]≥…≥c[1]≥c[0];
A=c[t]t;
B=c[t-1];
za svako i od t-2 do 0 uraditi:
        ako je A≥B, dopisati cifru c[i] broju B sa desne strane;
        u suprotnom, dopisati cifru c[i] broju A sa desne strane;
D=A * B;
rezultat=D mod 10^(broj cifara ispisa)
dopisati odgovarajući broj nula rezultatu sa leve strane
```

Nakon raspoređivanja cifara c_t i c_{t-1} važi $A \geq B$ i cifra c_{t-2} će biti dopisana broju B, a potom će cifra c_{t-3} biti dopisana broju A pošto je broj B veći jer ima veći broj cifara (napomena: A će biti veći ako je $c_t > 0$, a $c_{t-1} \geq 0$, ali tada su i sve cifre nadalje jednake 0 i svejedno je kom broju se dopisuju). Nakon raspoređivanja cifara c_{t-2} i c_{t-3}, brojevi A i B će imati isti broj cifara. Ako posmatramo naredne dve cifre (c_{t-4} i c_{t-5}), c_{t-4} će biti dopisana manjem (ili jednakom) broju, a c_{t-5} onom drugom broju. Na taj način, proces dopisivanja cifara možemo raditi u parovima (osim eventualno poslednje cifre).

Pogodno je da relaciju između brojeva A i B određujemo iterativno nakon svakog para cifara, dakle samo na osnovu dopisanih cifara. To možemo uraditi uvođenjem logičke promenljive jednaki koja ima vrednost true dok god su brojevi A i B jednaki, pri čemu vodimo računa da u slučaju nejednakosti uvek znamo koji je broj veći (recimo A).

Takođe, nije pogodno množenje brojeva A i B, jer oni mogu biti jako veliki i, s obzirom na to što je algoritam množenja velikih brojeva kvadratne složenosti poo broju njihovih cifara, vreme izvršenja ovog algoritma bilo bi jako veliko. Međutim, s obzirom na to što je potrebno odrediti samo zadnjih $M \leq 8$ cifara proizvoda, dovoljno je pomnožiti poslednjih M cifara brojeva A i B, odnosno naći poslednjih M cifara tog proizvoda.

Zgodno je, prilikom množenja, sve operacije obavljati po modulu 10^m. Direktno množenje brojeva $A_{M-1,0}$ i $B_{M-1,0}$ premašuje opseg 32-bitnog celobrojnog tipa. To nas navodi na množenje jednog od ova dva broja, recimo $A_{M-1,0}$, ciframa drugog broja, pri čemu se sve operacije obavljaju po modulu 10^m.

Konačno, algoritam se može napisati ovako:

```
sortirati niz cifara c u opadajući poredak c[t]≥c[t-1]≥…≥c[1]≥c[0];
A=c[t];
B=c[t-1];
ako je c[t]=c[t-1], jednaki=true;
u suprotnom, jednaki=false i veciJeA=true;
za svako i od t-2 do 0 sa korakom 2 uraditi:
```

```
                ako je  jednaki=true, uraditi sledeće:
                        dopisati cifru c[i] broju A sa desne strane;
                        ako je i-1≥0 uraditi:
                                dopisati cifru c[i-1] broju B sa desne strane;
                                ako je c[i]>c[i-1], jednaki=false;
                u suprotnom (jednaki=false),  uraditi sledeće:
                        dopisati cifru c[i] broju B sa desne strane;
                        ako je i-1≥0, dopisati cifru c[i-1] broju A sa desne strane;
        ako je n≥M-1,  ApoModulu=A[M-1,0];
        u suprtnom, ApoModulu=A;
        rezultat=0;
        za svako i od min(M-1, m) do 0 uraditi:
                rezultat=( 10 * rezultat+ApoModulu * B[i,i] ) mod 10^m
        dopisati odgovarajući broj nula rezultatu sa leve strane
```

fajl: proizvod.cpp

```
#include <fstream>
#include <vector>
#include <algorithm>

using namespace std;

int N, M;
int rezultat;
vector<int> nizCifara;
vector<int> prviBroj;
vector<int> drugiBroj;

void ucitavanje()
{
  int i;
  char ch;

  ifstream inFile = ifstream("proizvod.in");

  inFile >> N >> M;

  do
  {
    inFile >> ch;
  } while ((ch < '0') || (ch > '9'));

  nizCifara.push_back((int)ch - 48);

  for (i = 0; i < N - 1; i++)
  {
    inFile >> ch;
    nizCifara.push_back((int)ch - 48);
  }

  inFile.close();
}


void obrada()
{
  int i, j;
  int modul = 1;
  int prviBrojPoModulu;
  bool jednaki;

  for (i = 0; i < M; i++)
    modul *= 10;
```

```
  sort(nizCifara.begin(), nizCifara.end());

  prviBroj.push_back(nizCifara[N - 1]);
  drugiBroj.push_back(nizCifara[N - 2]);

  if (nizCifara[N - 1] == nizCifara[N - 2])
    jednaki = true;
  else
    jednaki = false;

  j = N - 3;

  // generisanje jednog i drugog broja od cifara
  while (j >= 0)
  {
    if (jednaki)
    {
      prviBroj.push_back(nizCifara[j]);
      j--;
      if (j >= 0)
      {
        drugiBroj.push_back(nizCifara[j]);
        if (nizCifara[j + 1] > nizCifara[j])
          jednaki = false;
        j--;
      }
    }
    else
    {
      drugiBroj.push_back(nizCifara[j]);
      j--;
      if (j >= 0)
      {
        prviBroj.push_back(nizCifara[j]);
        j--;
      }
    }
  }

  if (prviBroj.size() >= M)
    j = prviBroj.size() - M;
  else
    j = 0;

  prviBrojPoModulu = prviBroj[j];
  for (i = j + 1; i < prviBroj.size(); i++)
    prviBrojPoModulu = 10 * prviBrojPoModulu + prviBroj[i];

  if (drugiBroj.size() >= M)
    j = drugiBroj.size() - M;
  else
    j = 0;

  rezultat = 0;
  for (i = j; i < drugiBroj.size(); i++)
    rezultat = (10 * rezultat + prviBrojPoModulu * drugiBroj[i]) % modul;
}

void stampanje()
{
  ofstream outFile = ofstream("proizvod.out");

  int i;
  vector<int> cifreRezultata;

  while (rezultat > 0)
  {
```

```
    cifreRezultata.push_back(rezultat % 10);
    rezultat /= 10;
  }

  for (i = 0; i < M - cifreRezultata.size(); i++)
    outFile << 0;

  for (i = cifreRezultata.size() - 1; i >= 0; i--)
    outFile << cifreRezultata[i];

  outFile << endl;

  outFile.close();
}

int main()
{
  ucitavanje();
  obrada();
  stampanje();

  return 0;
}
```

fajl: proizvod2.cpp

```
#include <iostream>
#include <iomanip>
#include <string>
#include <algorithm>
#include <math.h>
using namespace std;

int main() {

    long long N, d[100000], M, k=0, A[2]={0,0}; string s; bool b=1;

    cin >> N >> M >> s;
    for (int i=0; i<N && ((d[i]=s[i]-'0')||1); i++) {}

    sort(d, d+N); reverse(d,d+N);

    for (long i=0; i<N && ((A[k] = (A[k]*10 + d[i]) % 1000000000L)||1); i++)
        b = !(!(k=1-k) && b && (d[i]!=d[i-1]) && (k=1-k)) && b;

    cout << setfill('0') << setw(M) << (A[0] * A[1] % (long long)pow(10.0,(double)M)) <<
endl;
}
```

zadatak: Prokopije i Simonid

Prokopije i Simonid su najbolji matematičari u razredu. Njihov drug Đurađ je zbog toga ljubomoran na njih pa im stalno smišlja neke smicalice. Ovoga puta je zamislio dva (ne obavezno različita) broja iz intervala [a,b] i saopštio Prokopiju njihov proizvod a Simonidu njihovu sumu (njih dvojica znaju iz kog intervala su odabrani ti brojevi). Prokopije i Simonid dalje vode sledeći razgovor:
P: „Ja ne znam koji su to brojevi".
S: „Ja ne znam koji su to brojevi".

P: „Ja ne znam koji su to brojevi".
S: „Ja ne znam koji su to brojevi".

.
.

.

P: „Ja ne znam koji su to brojevi".
S: „Ja ne znam koji su to brojevi".

P: „Sad znam koji su to brojevi".
Poznato je da su Prokopije i Simonid po n puta izrekli rečenicu „Ja ne znam koji su to brojevi", pre nego što je Prokopije saopštio da zna koji su to brojevi. Koje brojeve je zamislio Đurađ?

Ulaz:
U prvom i jedinom redu ulazne datoteke prosim.in se nalaze prirodni brojevi *a*, *b*, *n* ($1 \leq a < b \leq 1000$, $1 \leq n \leq 10$, $b - a \leq 128$) razdvojeni prazninom, koji označavaju, redom, početak i kraj intervala i broj ponavljanja goreoznačenog bloka u razgovoru.

Izlaz:
U prvom i jedinom redu izlazne datoteke **prosim.out** ispisati brojeve koje je Đurađ zamislio razdvojene prazninom, najpre manji pa veći (ukoliko su različiti).

Primer:

prosim.in	**prosim.out**
1 9 1	1 4

Objašnjenje:
Prokopiju je saopšten broj 4, Simonidu broj 5. Prokopije razmišlja: „Pošto je 4=1*4=2*2, ne mogu znati koje je brojeve zamislio Đurađ jer su obe kombinacije moguće", i kaže: P: „Ja ne znam koji su to brojevi". Simonid razmišlja: „Moguće su kombinacije 1, 4 kao i 2, 3. U prvom slučaju Prokopiju bi bio saopšten broj 4 i tada bi on bio u dilemi između kombinacija 1, 4 i 2, 2. U drugom slučaju bi mu bio saopšten broj 6 i bio bi u dilemi između kombinacija 1, 6 i 2, 3. Dakle, u oba slučaja bi Prokopije bio u dilemi (kao što i jeste), pa ne mogu znati o kom od njih se radi", i kaže: S: „Ja ne znam koji su to brojevi". Prokopije razmišlja: „Znam da je Simonidu saopšten ili broj 5 ili broj 4. Ukoliko bi to bio broj 4 on bi zaključio da su u pitanju brojevi 2, 2 jer bi u suprotnom meni bio saopšten broj 3 i odmah bih znao da je kombinacija 1, 3, ne bih bio u dilemi. Pošto Simonid ne zna koje je brojeve Đurađ zamislio ostaje jedino kombinacija 1, 4", i kaže: P: „Sad znam koji su to brojevi".

Napomena:
Garantuje se da će za sve test-primere rešenje biti jedinstveno.

Rešenje

Za svaki proizvod *p* u niz $pr_p.d_i$ pamtimo sume svake moguće kombinacije koja mu odgovara, a koja je prihvatljiva. Analogno za sume *s* pamtimo u niz $sum_s.d_i$. Na početku su sve kombinacije prihvatljive, a onda u svakoj od *n* iteracija iz svake sume izbacujemo one proizvode koji imaju jedinstvenu prihvatljivu sumu (njih smo ranije smestili u niz *zaup*). Istovremeno u niz *zaus* smeštamo one sume koje imaju jedinstven prihvatljiv proizvod, koje dalje analogno izbacujemo iz odgovarajućih proizvoda. Nakon završetka petlje imaćemo jedan proizvod u nizu *zaup*, i na osnovu njega i sume koje mu odgovara (jedinstvene) možemo lako izračunati zamišljenje brojeve.

Preudokod

```
za svaki proizvod/sumu formirati niz pr[p].d/sum[s].d svih odgovarajućih suma/proizvoda

formirati niz zaup[i] proizvoda koji imaju jedinstvenu prihvatljivu sumu

for k = 1 to n do
        for p in zaup
                izbaci(p,sum[pr[p].d[1]].d)
                if length(sum[pr[p].d[1]])=1
                        ubaci(pr[p].d[1],zaus)
                else
                        if length(sum[pr[p].d[1]])=0
                                izbaci(pr[p].d[1],zaus)
        for s in zaus
                izbaci(s,pr[sum[s].d[1]].d)
                if length(pr[sum[s].d[1]])=1
                        ubaci(sum[s].d[1],zaup)
                else
                        if length(pr[sum[s].d[1]].d)=0
                                izbaci(sum[s].d[1],zaup)
```

rešiti sistem jednacina x*y=zaup[1] i x+y=pr[zaup[1]].d[1]

traženi brojevi su x i y

fajl: prosim.pas

```
const gr=1000;
      int=128;
type matrp=record
            br:longint;
            d:array[1..gr] of byte;
           end;
     matrs=record
            br:longint;
            d:array[1..gr] of longint;
           end;
     tzau=record
           br:longint;
           d:array[1..gr*gr] of longint;
          end;
var a,b,n,i,j,k:longint;
    zaup,zaus:tzau;
    sum:array[0..2*int] of matrs;
    pr:array[0..gr*gr-(gr-int)*(gr-int)] of matrp;
    f:text;
procedure izbacis(u:longint;var niz:matrs);
begin
 j:=1;
 while ((niz.d[j]<>u)and(j<=niz.br)) do
    inc(j);
 if j<=niz.br then begin
                    niz.d[j]:=niz.d[niz.br];
                    dec(niz.br);
                   end;
end;
procedure izbacip(u:longint;var niz:matrp);
begin
 j:=1;
 while ((niz.d[j]<>u)and(j<=niz.br)) do
    inc(j);
 if j<=niz.br then begin
                    niz.d[j]:=niz.d[niz.br];
                    dec(niz.br);
                   end;
end;
procedure izbaci(u:longint;var niz:tzau);
begin
 j:=1;
 while ((niz.d[j]<>u)and(j<=niz.br)) do
    inc(j);
 if j<=niz.br then begin
                    niz.d[j]:=niz.d[niz.br];
                    dec(niz.br);
                   end;
end;
procedure skini_pr;
begin
 zaus.br:=0;
 for i:=1 to zaup.br do
   begin
    izbacis(zaup.d[i],sum[pr[zaup.d[i]-a*a].d[1]]);
    if sum[pr[zaup.d[i]-a*a].d[1]].br=1 then begin
                                              inc(zaus.br);
                                              zaus.d[zaus.br]:=pr[zaup.d[i]-a*a].d[1];
                                             end
                                        else if sum[pr[zaup.d[i]-a*a].d[1]].br=0 then
izbaci(pr[zaup.d[i]-a*a].d[1],zaus);
   end;
```

```
end;
procedure skini_sumu;
begin
 zaup.br:=0;
 for i:=1 to zaus.br do
   begin
    izbacip(zaus.d[i],pr[sum[zaus.d[i]].d[1]-a*a]);
    if pr[sum[zaus.d[i]].d[1]-a*a].br=1 then begin
                                              inc(zaup.br);
                                              zaup.d[zaup.br]:=sum[zaus.d[i]].d[1];
                                             end
                                       else if pr[sum[zaus.d[i]].d[1]-a*a].br=0 then
izbaci(sum[zaus.d[i]].d[1],zaup);

   end;
end;
begin
 assign(f,'prosim.in');
 reset(f);
 readln(f,a,b,n);
 close(f);
 for i:=a*a to b*b do
   pr[i-a*a].br:=0;
 for i:=2*a to 2*b do
   sum[i-2*a].br:=0;
 for i:=a to b do
   for j:=i to b do
     begin
      inc(pr[i*j-a*a].br);
      pr[i*j-a*a].d[pr[i*j-a*a].br]:=i+j-2*a;
      inc(sum[i+j-2*a].br);
      sum[i+j-2*a].d[sum[i+j-2*a].br]:=i*j;
     end;
 zaup.br:=0;
 for i:=a*a to b*b do
   if pr[i-a*a].br=1 then begin
                           inc(zaup.br);
                           zaup.d[zaup.br]:=i;
                          end;
 for k:=1 to n do
   begin
    skini_pr;
    skini_sumu;
   end;
 assign(f,'prosim.out');
 rewrite(f);
 for i:=1 to zaup.br do
   writeln(f,trunc(((pr[zaup.d[i]-a*a].d[1]+2*a)-sqrt((pr[zaup.d[i]-
a*a].d[1]+2*a)*(pr[zaup.d[i]-a*a].d[1]+2*a)-4*zaup.d[i]))/2),
            ' ',(pr[zaup.d[i]-a*a].d[1]+2*a)-
           trunc(((pr[zaup.d[i]-a*a].d[1]+2*a)-sqrt((pr[zaup.d[i]-
a*a].d[1]+2*a)*(pr[zaup.d[i]-a*a].d[1]+2*a)-4*zaup.d[i]))/2));
 close(f);
end.
```

fajl: prosim.cpp

```
/* Resenje takmicara Ivana Labata */


#include <cstdlib>
#include <iostream>

using namespace std;

class par
```

```
{
  public:
    par() {};
    par(int ax, int ay, int as): x(ax), y(ay), s(as), rem(false) {};
    inline bool operator==(const par& rhs) const
      {return (x == rhs.x && y == rhs.y);}
    inline bool operator<(const par& rhs) const
      {
        if (s < rhs.s) return true;
        if (s > rhs.s) return false;
        if (x < rhs.x) return true;
        if (x > rhs.x) return false;
        if (y < rhs.y) return true;
        return false;
      }
    int x,y,s;
    bool rem;
};

ostream & operator<< (ostream & o, const par & pr)
{
  char s[100];
  sprintf(s,"%(%d:%d,%d,%d)",pr.s,pr.x,pr.y,pr.rem);
  o << s;
}

int A = 0;
int B = 0;
int N = 0;
par * P;
par * Pn; //last+1
par * S;
par * Sn;


void
read_from_file(const char * filename)
{
  FILE * fi = fopen(filename,"r");
  fscanf(fi,"%d %d %d",&A,&B,&N);
  fclose(fi);
}

void
read_from_stdin()
{
  scanf("%d %d %d",&A,&B,&N);
}

inline void
add(int x, int y)
{
  Pn->x = x;
  Pn->y = y;
  Pn->s = x*y;
  Pn->rem = false;
  ++Pn;
  Sn->x = x;
  Sn->y = y;
  Sn->s = x+y;
  Sn->rem = false;
  ++Sn;
}

void
make()
{
```

```
  int n = (B-A+1)*(B-A+2)/2;
  P = new par[n+5];
  Pn = P;
  S = new par[n+5];
  Sn = S;
  for (int i=A; i<=B; ++i)
    for (int j=i; j<=B; ++j)
      add(i,j);
  sort(P,Pn);
  sort(S,Sn);
}

void
rem(par p, par * s, par * sn)
{
  par * f = lower_bound(s, sn, p);
  if (*f == p)
    f->rem = true;
  else
    cerr << "No match for " << p << "lower bound is " << *f << endl;
}

void
remove_pro()
{
  par * n = P;
  par * l = P;
  while ((n < Pn) && (n->rem))
    ++n;
  while (n < Pn)
  {
    par * f = n;
    int c = 0;
    for (; f->s == n->s; ++n)
      if (!n->rem)
        ++c;
    if (c == 1)
    {
      rem(par(f->x, f->y, f->x + f->y), S, Sn);
    }
    else if (c > 1)
    {
      while (f < n)
      {
        if (!f->rem)
          *l++ = *f++;
        else
          ++f;
      }
    }
    while ((n < Pn) && (n->rem))
      ++n;
  }
  Pn = l;
}

void
remove_sim()
{
  par * n = S;
  par * l = S;
  while ((n < Sn) && (n->rem))
    ++n;
  while (n < Sn)
  {
    par * f = n;
    int c = 0;
```

```
    for (; f->s == n->s; ++n)
      if (!n->rem)
        ++c;
    if (c == 1)
    {
      rem(par(f->x, f->y, f->x * f->y), P, Pn);
    }
    else if (c > 1)
    {
      while (f < n)
      {
        if (!f->rem)
          *l++ = *f++;
        else
          ++f;
      }
    }
    while ((n < Sn) && (n->rem))
      ++n;
  }
  Sn = l;
}

void
find_pro()
{
  par * n = P;
  while ((n < Pn) && (n->rem))
    ++n;
  while (n < Pn)
  {
    par * f = n;
    int c = 0;
    for (; f->s == n->s; ++n)
      if (!n->rem)
        ++c;
    if (c == 1)
    {
      printf("%d %d\n",f->x, f->y);
    }
    while ((n < Pn) && (n->rem))
      ++n;
  }
}


void
solve()
{
  for (int i=0; i<N; ++i)
  {
    remove_pro();
    remove_sim();
  }
  find_pro();
}

int
main(int argc, char *argv[])
{
  if (argc-1 >= 1)
    read_from_file(argv[1]);
  else
    read_from_stdin();
  make();
  solve();
  return EXIT_SUCCESS;
```

```
}
```

zadatak: Bušene kartice

Profesor Đurić veoma voli da programira. Međutim, kako je uvek veoma zauzet, nije stigao da nauči ni jedan moderan programski jezik već još uvek programira bušeći kartice. To radi tako što uzme jednu praznu karticu (bez rupa) i potom buši jednu po jednu rupu, pri tome praveći sigurnosne provere nakon svake probušene rupe. Svaku sigurnosnu proveru profesor izvodi na sledeći način: Najpre načini identičnu kopiju kartice na kojoj radi. Potom tu kopiju stavi iznad originalne kartice tako da im se sve rupe poklope, a onda počne da je pomera, pri čemu pazi da je ne zarotira. Provera traje dok ne isproba sve moguće položaje gornje kartice u odnosu na donju. Rezultat provere je najveći broj rupa koje su se istovremeno poklopile (ne računajući početni položaj kada se sve rupe poklapaju). Pošto su sigurnosne provere profesoru dosadne za izvođenje, zamolio je vas da u nekom malo savremenijem programskom jeziku napišete program koji nalazi rezultate svih sigurnosnih provera.

Ulaz:

Ulazni podaci se učitavaju iz tekstualnog fajla **kartice.in**. U prvoj liniji ulaza nalazi se prirodan broj n, ukupan broj rupa koji profesor treba da probuši ($1 \leq n \leq 3000$). U svakom od narednih n redova nalaze se dva razmakom razdvojena cela broja x i y, koji predstavljaju koordinate rupe ($-2^{30} < x, y < 2^{30}$). Rupe su date redom kojim ih profesor buši. Ne postoje dve rupe sa istim koordinatama.

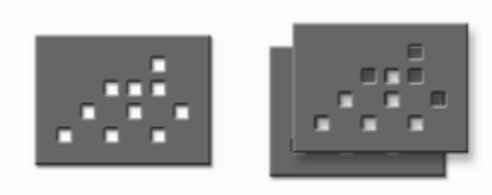
Izlaz:

U izlazni tekstualni fajl **kartice.out** treba zapisati rezultate svih n sigurnosnih provera, u svakoj liniji po jedan, redom kojim su se provere izvodile.

Primer:

kartice.in	kartice.out
10	0
5 1	1
4 2	1
3 1	2
3 3	3
2 2	3
1 1	3
4 3	4
5 3	5
5 4	6
6 2	

Objašnjenje:



Na slici je prikazan izgled kartice nakon svih deset probušenih rupa i položaj kartica u sigurnosnoj proveri u kome ima najviše poklopljenih rupa.

Rešenje

Označimo sve rupe brojevima od 1 do n po redu kojem ih je profesor Đurić bušio, a sa $p[i]$ vektor koordinate i-te tačke.

Pretpostavimo da je profesor upravo izbušio k-tu rupu, i da treba odrediti rezultat sigurnosne provere za prvih k rupa.

Ako se gornja kartica translira za vektor t u odnosu na donju, doći će do poklapanja rupe i sa donje kartice i rupe j sa gornje ako i samo ako je $p[i] - p[j] = t$. Iz toga sledi da je maksimalan broj preklapanja jednak najvećem broju ponavljanja nekog vektora među svim vektorima iz multiskupa $T_k = \{p[i] - p[j] : 1 \leq i, j \leq k\}$

Ovo možemo brojati na više načina, a najefikasnije je da se multiskup predstavi pomoću heš tabele, i prilikom svake nove probušene rupe proširuje novim vektorima (vektori između nove i već postojećih rupa).

Obratite pažnju da nije potrebno pamtiti sve vektore, jer vektori t i $-t$ uvek dolaze u paru, pe je dovoljno pamtiti samo jedan od njih. Zato uvedimo relaciju totalnog poretka među vektorima takvu da je $a \leq b$ ako i samo ako važi (($a.x < b.x$) or (($a.x = b.x$) and ($a.y \leq b.y$))). Tada kada god posmatramo dve tačake $p[i]$ i $p[j]$ ubacićemo onaj od vektora $p[i] - p[j]$ i $p[j] - p[i]$ koji je veći od nula vektora.
Ako se napiše dobra heš funkcija, složenost rešenja je $O(n^2)$. Jedino je ovakvo rešenje moglo doneti maksimum bodova.
Zadatak je moguće rešiti i na nekoliko načina u vremenu $O(n^2 \lg n)$.
Jedan od načina je da vektore koje formiraju sve rupe najpre sortiramo u niz i izbacimo duplikate. Potom idemo po rupama od prve probušene do poslednje, i za svaku nalazimo sve vektore koje ona formira sa do sada probušenim rupama. Sve te vektore binarnom pretragom nađemo u nizu i povećamo njima pridružen brojač za jedan.

fajl: kartice_hash.pas

```
const
  fin = 'kartice.in';
  fout = 'kartice.out';
  MaxN = 3000;


type
  TVector = record
    x, y : Longint;
  end;

  PHashEntry = ^THashEntry;
  THashEntry = record
    v : TVector;
    count : Integer;
    next : PHashEntry;
  end;


const
  zero : TVector = (x : 0; y : 0);
  hashSize = 19260479;


var
  n : Integer;
  p : array[1..MaxN] of TVector;

  sol : array[1..MaxN] of Integer;
  max : Integer;

  hash : array[0..HashSize-1] of PHashEntry;



  function Compare(const a, b : TVector) : Longint;
  begin
    Compare := a.x - b.x;
    if Compare = 0 then
      Compare := a.y - b.y;
  end;


  function Sub(const a, b : TVector) : TVector;
  begin
    Sub.x := a.x - b.x;
    Sub.y := a.y - b.y;
  end;


  function Rem(a, b : Longint) : Longint;
```

```
begin
  if a > 0 then
    Rem := a mod b
  else
  begin
    Rem := (-a) mod b;
    if Rem > 0 then
      Rem := b - Rem;
  end;
end;


procedure Insert(const t : TVector);
var
  k : Longint;
  p : PHashEntry;
begin
  k := (7333 * Rem(t.x, 98467) + 9839 * Rem(t.y, 89783)) mod hashSize;

  p := hash[k];
  while (p <> nil) and (Compare(p^.v, t) <> 0) do
    p := p^.next;

  if p = nil then
  begin
    p := New(PHashEntry);
    p^.count := 0;
    p^.v := t;
    p^.next := hash[k];
    hash[k] := p;
  end;

  inc(p^.count);
  if p^.count > max then
    max := p^.count;
end;


procedure Solve;
var
  i, j : Integer;
  t, h : TVector;
  k : Longint;
begin
  for k := 0 to hashSize - 1 do
    hash[k] := nil;

  max := 0;
  for i := 1 to n do
  begin
    for j := 1 to i-1 do
    begin
      t := Sub(p[i], p[j]);
      if Compare(t, zero) < 0 then
        t := Sub(zero, t);

      Insert(t);
    end;
    sol[i] := max;
  end;
end;


procedure ReadInput;
var
  f : Text;
  i : Integer;
```

```
  begin
    Assign(f, fin);
    Reset(f);
    Readln(f, n);
    for i := 1 to n do
      Readln(f, p[i].x, p[i].y);
    Close(f);
  end;


  procedure WriteOutput;
  var
    f : Text;
    i : Integer;
  begin
    Assign(f, fout);
    Rewrite(f);
    for i := 1 to n do
      Writeln(f, sol[i]);
    Close(f);
  end;



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

FAJL: KARTICE_BINARY_SEARCH.PAS

```
const
  fin = 'kartice.in';
  fout = 'kartice.out';
  MaxN = 3000;
  MaxM = (MaxN * (MaxN - 1)) DIV 2;


type
  TVector = record
    x, y : Longint;
  end;

  TListEntry = record
    v : TVector;
    count : Integer;
  end;


const
  zero : TVector = (x : 0; y : 0);


var
  n : Integer;
  p : array[1..MaxN] of TVector;

  max : Integer;
  sol : array[1..MaxN] of Integer;

  m : Longint;
  list : array[1..MaxM] of TListEntry;
```

```
function Compare(const a, b : TVector) : Longint;
begin
  Compare := a.x - b.x;
  if Compare = 0 then
    Compare := a.y - b.y;
end;


function Sub(const a, b : TVector) : TVector;
begin
  Sub.x := a.x - b.x;
  Sub.y := a.y - b.y;
end;


procedure QuickSort;
var
  p : TVector;
  tmp : TListEntry;

  procedure Sort(l, r : Longint);
  var
    i, j : Longint;
  begin
    i := l;
    j := r;
    p := list[(l + r) DIV 2].v;

    repeat
      while Compare(list[i].v, p) < 0 do inc(i);
      while Compare(list[j].v, p) > 0 do dec(j);
      if i <= j then
      begin
        tmp := list[i];
        list[i] := list[j];
        list[j] := tmp;
        inc(i);
        dec(j);
      end;
    until i > j;

    if (l < j) then Sort(l, j);
    if (i < r) then Sort(i, r);
  end;

begin
  Sort(1, m);
end;


procedure Solve;
var
  i, j : Integer;
  t : TVector;
  k, l : Longint;
  a, b, mid : Longint;
  c : Longint;
  found : Boolean;
begin
  m := 0;

  for i := 1 to n - 1 do
    for j := i + 1 to n do
    begin
      t := Sub(p[i], p[j]);
```

```
      if Compare(t, zero) < 0 then
        t := Sub(zero, t);

      inc(m);
      list[m].v := t;
      list[m].count := 0;
    end;

  QuickSort;

  l := 1;

  for k := 2 to m do
    if Compare(list[k].v, list[l].v) <> 0 then
    begin
      inc(l);
      list[l] := list[k];
    end;

  max := 0;

  for i := 1 to n do
  begin
    for j := 1 to i-1 do
    begin
      t := Sub(p[i], p[j]);
      if Compare(t, zero) < 0 then
        t := Sub(zero, t);

      a := 1;
      b := l;

      found := false;
      while not found do
      begin
        mid := (a + b) div 2;

        c := Compare(list[mid].v, t);
        if c < 0 then
          a := mid + 1
        else
          if c > 0 then
            b := mid - 1
          else
            found := true;
      end;

      inc(list[mid].count);
      if list[mid].count > max then
        max := list[mid].count;
    end;
    sol[i] := max;
  end;
end;


procedure ReadInput;
var
  f : Text;
  i : Integer;
begin
  Assign(f, fin);
  Reset(f);
  Readln(f, n);
  for i := 1 to n do
    Readln(f, p[i].x, p[i].y);
  Close(f);
```

```
  end;


  procedure WriteOutput;
  var
    f : Text;
    i : Integer;
  begin
    Assign(f, fout);
    Rewrite(f);
    for i := 1 to n do
      Writeln(f, sol[i]);
    Close(f);
  end;



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

zadatak: Planete

Patak Dača je dobio novi zadatak da prati planete po zadatom redosledu. Ukupno ima 26 planeta i one su obeležene malim slovima engleskog alfabeta. Patak Dača, međutim, ume da pobrka planete, pa se na kraju ispostavi da niz planeta koje je on obišao ne odgovara nizu planeta koji je stajao u njegovom zadatku. Svemirska komisija utvrđuje kaznene poene tako što nalazi minimalan broj korekcija putanje potreban da se niz planeta koje je Dača obišao prevede u niz planeta u zadatku. Jednu korekciju putanje može predstavljati: (1) zamena jedne planete drugom, (2) umetanje jedne planete, (3) izbacivanje jedne planete, i (4) zamena mesta dveju planeta koje su susedne u početnom nizu (tj. koje su bile susedne i pre svih korekcija). Vaš zadatak je da pomognete komisiji i izračunate broj kaznenih poena koji se dodeljuju Dači, znajući da on, zahvaljujući svom umeću, nikada ne dobija više od 100 kaznenih poena.
Za ovaj zadatak potrebno je da predate izlazne datoteke za 10 ulaznih datoteka pod nazivom **planete.01.in**, …, **planete.10.in** koje se nalaze u arhivi koja vam je data. Izlazne datoteke je potrebno nazvati imenima **planete.01.out**, …, **planete.10.out**, pri čemu broj u nazivu izlazne datoteke odgovara broju u nazivu ulazne datoteke za dati test primer. Izlazne datoteke treba priložiti (submitovati).
Ulaz:
U prvoj liniji ulaznog tekstualnog fajla nalaze se dva prirodna broja *N* i *M* razdvojenih blanko znakom, koji predstavljaju broj planeta u zadatku i broj planeta koje je Patak Dača obišao, respektivno. U drugoj liniji nalazi se *N* malih slova engleskog alfabeta koje predstavljaju niz planeta u zadatku koji je postavljen Dači. Pre i između slova u drugoj liniji ne postoji ni jedan blanko znak. U trećoj liniji nalazi se *M* malih slova engleskog alfabeta koje predstavljaju niz planeta koje je Dača obišao. Pre i između slova u trećoj liniji ne postoji ni jedan blanko znak.
Izlaz:
U prvoj liniji izlaznog tekstualnog fajla treba zapisati "`# planete, nn`" (bez navodnika) gde umesto *nn* treba zapisati broj test primera. U drugoj liniji izlaznog tekstualnog fajla upisati broj kaznenih poena *K* koje je Dača zaradio. U svakoj od narednih *K* linija upisati po jednu korekciju, tako da se njihovom primenom u datom redosledu niz planeta koje je Dača obišao može prevesti u niz planeta koje stoje u opisu njegovog zadatka. Za korekciju (1) ispis se vrši u formatu: 1 *H Y*, pri čemu *H* predstavlja redni broj planete koja se zamenjuje, a *Y* oznaku nove planete na toj poziciji. Za korekciju (2) ispis se vrši u formatu: 2 *H Y*, pri čemu *H* predstavlja redno mesto u nizu na koji se umeće planeta, a *Y* oznaku planete koja se umeće. Za korekciju (3) ispis se vrši u formatu: 3 *H*, pri čemu *H* predstavlja redni broj planete koja se izbacuje. Za korekciju (4) ispis se vrši u formatu: 4 *H*, pri čemu *H* predstavlja redni broj planete koja se zamenjuje sa sledećom. Ukoliko postoji veći broj sekvenci korekcija koje predstavljaju rešenje, štampati bilo koju od njih.
Primer:

planete.00.in	**planete.00.out**
`8 8`	`# planete, 00`
`computer`	`4`
`kmpjutre`	`1 1 c`
	`2 2 o`

```
3 5
4 7
```

Objašnjenje:
Minimalan broj korekcija putanje je 4. U prvoj korekciji, prva planeta (`k`) se zamenjuje planetom `c`, pa se dobija niz: `kmpjutre` -> `cmpjutre`. U drugoj korekciji, na drugu poziciju umeće se planeta `o`, pa se dobija niz: `cmpjutre` -> `compjutre`. U trećoj korekciji, izbacuje se planeta sa pete pozicije, pa se dobija niz: `compjutre` -> `computre`. U četvrtoj korekciji, zamenjuju se planete na sedmoj i osmoj poziciji, pa se dobija niz: `computre` -> `computer`

fajl: planete.cpp

```
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

int N, M, Krez, pomeraj;
vector<char> put1, put2;
vector<int> transfList[100001][101];
ofstream outFile("planete.00.out");

void ucitavanje()
{
  int i;
  char ch;

  ifstream inFile("planete.00.in");

  inFile >> N >> M;

  do
  {
    inFile >> ch;
  } while ((ch < 'a') || (ch > 'z'));

    put2.push_back(ch);

  for (i = 0; i < N - 1; i++)
  {
    inFile >> ch;
    put2.push_back(ch);
  }

  do
  {
    inFile >> ch;
  } while ((ch < 'a') || (ch > 'z'));

  put1.push_back(ch);

  for (i = 0; i < M - 1; i++)
  {
    inFile >> ch;
    put1.push_back(ch);
  }

  inFile.close();
}

bool postoji(int i, int j, int k)
{
```

```
  bool already = false;
  int l = 0;
  while ((!already) && (l < transfList[i][j].size()))
  {
    if ((transfList[i][j][l] >> 3) == k)
      already = true;
    l++;
  }
  return already;
}

int obrada()
{

  int i, j, k, l, t;
  bool found1, found2;

  transfList[0][0].push_back(0);

  /// main loop
  for (k = 0; k <= 100; k++)
  {
    found1 = false;

    found2 = false;

    cout << k << endl;

    for (i = 0; i <= put1.size(); i++)
      for (l = 0; l < transfList[i][k].size(); l++)
      {
        j = transfList[i][k][l] >> 3;
        /// brisanje i + 1
        if (i < put1.size())
          if (k + 1 + abs(put1.size() - (i + 1) - put2.size() + j) <= 100)
            if (!postoji(i + 1, k + 1, j))
            {
              transfList[i + 1][k + 1].push_back((j << 3) | 1);
              if ((i + 1 == put1.size()) && (j == put2.size()))
                found2 = true;
            }
        /// dodavanje izmedju i i i + 1
        if (j < put2.size())
          if (k + 1 + abs(put1.size() - i - put2.size() + j + 1) <= 100)
            if (!postoji(i, k + 1, j + 1))
            {
              transfList[i][k + 1].push_back(((j + 1) << 3) | 2);
              if ((i == put1.size()) && (j + 1 == put2.size()))
                found2 = true;
            }
        /// zadrzavanje ili promena i + 1
        if ((i < put1.size()) && (j < put2.size()))
          /// zadrzavanje i + 1
          if (put1[i] == put2[j])
          {
            if (!postoji(i + 1, k, j + 1))
            {
              transfList[i + 1][k].push_back(((j + 1) << 3) | 3);
              if ((i + 1 == put1.size()) && (j + 1 == put2.size()))
                found1 = true;
            }
          }
          /// promena i + 1
          else
          {
            if (k + 1 + abs(put1.size() - i - put2.size() + j) <= 100)
              if (!postoji(i + 1, k + 1, j + 1))
```

```
              {
                transfList[i + 1][k + 1].push_back(((j + 1) << 3) | 4);
                if ((i + 1 == put1.size()) && (j + 1 == put2.size()))
                  found2 = true;
              }
          }
        /// zamena i + 1 i i + 2
        if ((i + 1 < put1.size()) && (j + 1 < put2.size()))
          if ((put1[i] == put2[j + 1]) && (put1[i + 1] == put2[j]) && (put1[i] != put1[i +
1]))
            if (k + 1 + abs(put1.size() - i - put2.size() + j) <= 100)
              if (!postoji(i + 2, k + 1, j + 2))
              {
                transfList[i + 2][k + 1].push_back(((j + 2) << 3) | 5);
                if ((i + 2 == put1.size()) && (j + 2 == put2.size()))
                  found2 = true;
              }
      }

    if (found1)
      return k;
    else
      if (found2)
      return k + 1;
  }

  return -1;
}

void stampanje(int i, int k, int j)
{
     int l, c;
     bool ponovi = true;

     while (ponovi)
     {
         ponovi = false;

     l = 0;
         while (l < transfList[i][k].size())
         {
               if ((transfList[i][k][l] >> 3) == j)
                  break;
               else
                   l++;
         }

         c = transfList[i][k][l] & 7;

         switch (c)
         {
                case 1 :
                     stampanje(i - 1, k - 1, j);
                     outFile << "3 " << i + pomeraj << endl;
                     pomeraj--;
                     break;

                case 2 :
                     stampanje(i, k - 1, j - 1);
                     outFile << "2 " << i + 1 + pomeraj << " " << put2[j - 1] << endl;
                     pomeraj++;
                     break;

                case 3 :
                     i--;
                     j--;
                     ponovi = true;
```

```
                    break;

                case 4 :
                    stampanje(i - 1, k - 1, j - 1);
                    outFile << "1 " << i + pomeraj << " " << put2[j - 1] << endl;
                    break;

                case 5 :
                    stampanje(i - 2, k - 1, j - 2);
                    outFile << "4 " << i - 1 + pomeraj << endl;
            }
        }
}

int main()
{
  ucitavanje();
  Krez = obrada();
  outFile << Krez << endl;
  pomeraj = 0;
  if (Krez > -1)
       stampanje(put1.size(), Krez, put2.size());
  outFile.close();

  return 0;
}
```

zadatak: Reli

U zemlji Bajtoviji se, tradicionalno, svake prestupne godine održava egzibiciona reli trka. Vozači i ekipe se okupljaju u Donjem Bitu, gde izlažu svoje automobile zainteresovanoj publici. Zatim svaki vozač dobija startni broj i trka počinje. Vozači kreću ka Gornjem Bitu prema startnim brojevima u razmacima od po 3 minuta. Po završetku trke, u Gornjem Bitu se održava veliko slavlje nakon koga publika i učesnici kolektivno peru sudove. Tek nakon što su sudovi oprani, proglašava se pobednik trke. Legenda kaže da će zavladati glad do sledećeg relija, ako se pobednik proglasi uz prljave sudove.
I tako, od pamtiveka...
Uz savremeno doba dođoše i savremena shvatanja. Ministarstvo Saobraćaja, Automobilizma i Turizma, smatra da su trke suviše opasne i želi da ih zabrani uz obrazloženje da raspon od 3 minuta ostavlja veoma male manevarske mogućnosti i dovodi do neverovatno velikog broja jako rizičnih preticanja. Ako bi se interval povećao na bezbednih 7 minuta, trke bi trajale do kasno u noć kada više niko ne bi ostao da pere sudove. Potrebno je uveriti predstavnike Ministarstva da i pored raspona od 3 minuta ne dolazi do prevelikog broja preticanja.
Poznati su vam podaci vezani za poslednju održanu trku, preciznije: broj učesnika N ($1 \leq N \leq 100000$) i redosled kojim su takmičari stizali u Gornji Bit. Od vas se očekuje da odredite minimalan broj preticanja koji se mogao dogoditi na datoj trci.

Napomena:
Automobili takmičara su obeleženi startnim brojevima od 1 do N, i tim redom kreću sa starta. U prilog trci ide i podatak da u svakom preticanju učestvuju samo dva vozača. Još se nije desilo da neko od takmičara ne završi trku!

Ulaz:
U prvoj liniji tekstualnog fajla reli.in se nalazi broj vozača N. U svakoj od sledećih N linija se nalazi po jedan broj, i to u i+1-oj liniji, startni broj vozača koji je i-ti stigao u Gornji Bit.

Izlaz:
U prvoj i jedinoj liniji tekstualnog fajla **reli.out** potrebno je ispisati ostatak minimalnog broja preticanja pri deljenju sa 10000.

Primer:

reli.in	reli.out
5 4 3 2 1	

5

Rešenje

Očigledno je da se u zadatku trazi broj zamena mesta susednih elemenata potrebnih za sortiranje datog niza.
Trivijalno resenje koristi *BubbleSort* algoritam i time simulira ceo proces. Vremenska slozenost ovakvog resenja je $O(n^2)$ i ono donosi oko 20% poena.
Jedno od efikasnijih resenja koristi modifikaciju *MergeSort* algoritma. *MergeSort* radi tako to niz podeli na dva podniza približno jednakih dužina, sortira levi i desni podniz a zatim elemente podnizova raspoređuje tako da se dobije sortiran niz. Pod pretpostavkom da su podnizovi sortirani, *MergeSort* bira manji od čeonih elemenata dva podniza, stavlja ga na kraj novog niza i uklanja iz odgovarajućeg podniza. Funkcija se izvrsava sve dok u bar jednom podnizu ima elemenata. Niz dobijen ovim postupkom je sortiran, čime je zadovoljena pretpostavka. Broj zamena potrebnih za sortiranje niza jednak je zbiru broja zamena potrebnih za sortiranje dva podniza i broja zamena potrebnih za raspoređivanje elemenata. Kako se raspoređivanje vrši linearno, dokazuje se da je vremenska složenost ovakvog algoritma $O(n \lg n)$.

fajl: reli.cpp

```
/*
ZADATAK: reli
JEZIK: cpp
*/
#include <fstream.h>
#include <iostream.h>
#include <math.h>

const long int MAXN = 200000;
long int n, a[MAXN], b[MAXN];

char* infile = "reli.in";
char* outfile = "reli.out";

long int joe_100(long int left, long int right);

void save()
{
  ofstream fout(outfile);

  fout << joe_100(1, n) << endl;

}

void load()
{
  ifstream fin(infile);

  fin >> n;

  for (long int i = 1; i <= n; i++)
  {
    fin >> a[i];

  }

  a[n + 1] = n + 2;
}



long int joe_100(long int left, long int right)
{
  if (left == right) return 0;
```

```
  long int m = (left + right) / 2;

  long int flips = joe_100(left, m) + joe_100(m + 1, right);

  long int i = left;
  long int j = m + 1;
  long int k = left;
  long int sub = 0;

  while ((i <= m) && (j <= right))
  {
    if (a[i] < a[j])
    {
      b[k] = a[i];
      sub += abs(i - k);
      sub = sub % 100000;
      i++;
      k++;
    }
    else

    {
      b[k] = a[j];
      sub += abs(j - k);
      sub = sub % 100000;
      j++;
      k++;
    }
  }

  while (i <= m)
  {
    b[k] = a[i];
    sub += abs(i - k);
    sub = sub % 100000;
    i++;
    k++;
  }

  while (j <= right)
  {
    b[k] = a[j];
    sub += abs(j - k);
    sub = sub % 100000;
    j++;
    k++;
  }

  for (i = left; i <= right; i++) a[i] = b[i];

  flips += sub / 2;

  return (flips % 10000);
}


int main()
{
  load();
  save();

  return 0;
}
```

fajl: reli2.cpp

```
#include <iostream>
#include <vector>

using namespace std;

long Count(vector<long> &A){

  if (A.size() <= 1)
    return 0;

  vector<long> A1, A2;

  for (long i=0;i<A.size()/2;i++)
    A1.push_back(A[i]);
  for (long i=A.size()/2;i<A.size();i++)
    A2.push_back(A[i]);

  long m = 0, c = Count(A1) + Count(A2);

  A.clear();
  for (long i=0;i<A1.size();i++) {
    for (;(m<A2.size())&&(A2[m]<A1[i]);m++) {
      A.push_back(A2[m]);
      c+= A1.size()-i;
    }
    A.push_back(A1[i]);
  }

  for (;m<A2.size();m++)
    A.push_back(A2[m]);

  return c;
}

int main(){

  long N, tmp;

  cin >> N;
  vector<long> A(N);

  for (long i=0;i<N;i++)
    cin >> A[i];

  cout << Count(A);
}
```

zadatak: Poker

Biznismen Draganče je odlučio da poveća prihode svog kazina tako što će organizovati poker turnire sa milionskim nagradnim fondom. Zato se obratio svom drugu Đuri, direktoru najgledanije televizije koji se odmah oduševio idejom brojeći nove gledaoce koje će ovaj program doneti. Kada je shvatio koliki je iznos licence za takvu emisiju odlučio je da malo promeni pravila. Međutim ovo im je donelo jedan problem: kako izračunati verovatnoću da određeni igrač pobedi samo na osnovu prve dve podeljene karte, što je najinteresantniji detalj za televizijske gledaoce. Pomozite im da reše ovaj problem.
Pravila njihove verzije televizijskog pokera su sledeća: Igra se sa 32 karte, po 8 (A K Q J 10 9 8 7) od svake od 4 boje (P K H T). Karte poređane po jačini: A K Q J 10 9 8 7. Svakom igraču se na početku podele dve karte, a zatim se na sto stavi još pet karata. Svaki igrač bira pet karata od sledećih sedam: dva iz ruke i pet sa stola, tako da one čine najjaču moguću kombinaciju. Na kraju igrač koji je imao najjaču kombinaciju pobeđuje, a ako su dva ili više igrača izjednačeni onda je rezultat partije nerešen. Kombinacije su poređane po jačini na sledeći način, a dodatne napomene date su u zagradama:

1. fleš rojal: A,K,Q,J,10 u istoj boji (sve boje su ravnopravne) (primer: PA, PK, PQ, PJ, P10)
2. boja-kenta: pet karata iste boje poređane po redu (najjača karta odlučuje) (primer: HJ, H10, H9, H8, H7)
3. poker: četiri karte iste vrste (ako dva igrača imaju poker, pobednik je onaj ko je ima poker sastavljen od karata jače vrste) (primer: PA, KA, HA, TA, T8)
4. ful: tri karte jedne vrste i dve druge vrste (prvo se gleda jačina tri karte, a onda dve karte) (primer: PK, KK, TK, K7, T7)
5. boja: pet karata iste boje (najjača karta odlučuje) (primer: PQ, PJ, P10, P8, P7)
6. kenta: pet karata poređane po redu (najjača karta odlučuje) (primer: TQ, HJ, T10, H9, K8)
7. triling: tri karte iste vrste (ako dva igrača imaju triling, pobeđuje onaj koji ima triling od karata jače vrste) (primer: P10, K10, T10, K7, T9)
8. dva para: dve karte jedne vrste i dve karte druge vrste (prvo se gleda jači, pa slabiji par) (primer: HA, TA, T8, P8, PJ)
9. jedan par: dve karte iste vrste (ako dva igrača imaju po jedan par, pobednik je onaj koji ima par od jačih karata) (primer: T8, P8, K7, TQ, HA)

Napomena:
Verovatnoća da određeni igrač pobedi dobija se deljenjem broja ishoda u kojima taj igrač pobeđuje sa ukupnim brojem ishoda. Jedan špil sadrži po jednu od prethodno opisane 32 karte.
Ulaz:
U prvoj liniji ulaznog tekstualnog fajla **poker.in** nalazi se prirodan broj N ($2 \leq N \leq 10$) , broj igrača. U sledećih $2N$ linija nalaze se karte koje su podeljene igračima na početku i to u drugoj i trećoj karte prvog igrača, u četvrtoj i petoj drugog... Sve karte su date u obliku boja razmak vrsta, gde su boje označene sa P K H T, a vrste sa A K Q J 10 9 8 7.
Izlaz:
U prvoj liniji izlaznog tekstualnog fajla **poker.out** ispisati verovatnoću da će pobediti prvi igra, u drugoj drugi... U N+1-voj liniji ispisati verovatnću da će biti nerešeno. Sve verovatnoće iskazati u procentima. Dozvoljeno odstupanje od tačne vrednosti je 0.01.
Primer:

poker.in	**poker.out**
3 P A H A T A T 8 T 7 H 8	

Rešenje

Zadatak se rešava pravolinijskom simulacijom. Generišu se sve mogućnosti i onda se ispita ishod svake. Međutim, postoje detalji na koje se mora paziti.
Prvi od njih je predstavljanje podataka. Iako se karte mogu pamtiti kao stringovi to je veoma neprakticno zbog poređenja. Mnogo je lakše kodirati ih kao brojeve, poredane po jačini. Pošto od svake vrste ima po 8 karata, A će biti predstavljen sa 7, K sa 6 ... 7 sa 0. Radi lakšeg generisanja kombinacija sve karte ce biti poređane po bojama i kodirane brojevima od 0 do 31. Na primer K 10 ce biti broj 11.
Kombinacije se generišu standardnim algoritmom za generisanje kombinacija bez ponavljanja. Od nepodeljenih karata se bira pet i na onda se računa ishod partije ako su tih pet karata na stolu.
Sad dolazimo do drugog pipavog detalja: kako odrediti pobednika, odnosno kako porediti najbolje kombinacije karata za svakog igrača. Mogli bi napisati po jednu proceduru za svaku kombinaciju karata (par, dva para...) i onda za proveravati za svaku od tih karata da li neki igrač može da je sastavi od svoje dve i pet karata sa stola. Međutim, mnogo je praktičnije za svakog igraca odrediti najbolji ishod i samo njih porediti. Pamćenje tih ishoda se najlakše realizuje kodiranjem u prirodan broj tako da veci broj pokazuje bolji ishod. Na kraju se najbolji ishodi svakog igraca porede i određuje pobednik čiji se broj pobeda poveća za 1. Ako je nerešeno povećava se broj nerešenih partija. Takođe se broji i ukupan broj partija.
Na kraju se u izlazni fajl ispišu verovatnoće koje se dobijaju kad se broj pobeda svakog igrača pomnožen sa 100 podeli sa ukupnim brojem partija.
Prilažemo dva rešenja, pri čemu drugo rešenje (pisano u paskalu) radi znatno sporije, jer proveru radi za sve petočlane podskupove skupa od 7 karata.

fajl: poker.cpp

```
#include <fstream>
#include <string>
using namespace std;
void init(bool karte[])
{
  for (int i=0;i<32;i++)
    karte[i]=false;
}
bool karte[32],sto[32];
bool igrac[10][32];
int ukupno,nereseno,n;
int dobio[10];
ifstream fin("poker.in");
ofstream fout("poker.out");


int skor(int brojIgraca)
{
  int i,j,k;
  bool pom[32];
  int boja[4];
  for (i=0;i<4;i++) boja[i]=0;
  int vrsta[8];
  for (j=0;j<8;j++) vrsta[j]=0;
  for (i=0;i<32;i++) pom[i]=false;
  for (i=0;i<4;i++)
    for (j=0;j<8;j++)
    if (igrac[brojIgraca][8*i+j]||sto[8*i+j])
    {
      boja[i]++;  //brojimo karte iste boje
    vrsta[j]++;  //brojimo karte iste vrste
    pom[8*i+j]=true;
      }
  for (i=0;i<4;i++)
    if (pom[8*i+7]&&pom[8*i+6]&&pom[8*i+5]&&pom[8*i+4]&&pom[8*i+3])
    return 800; //fles rojal
  for (i=0;i<4;i++)
    if (boja[i]>=5)
    {
      for (j=6;j>3;j--)
      if (pom[8*i+j]&&pom[8*i+j-1]&&pom[8*i+j-2]&&pom[8*i+j-3]&&pom[8*i+j-4])
        return 700+j; //fles, j najjaca karta
    for (j=7;j>=0;j--)
      if (vrsta[j]==4)
        return 600+j; //poker
      for (j=7;j>=0;j--)
      for (k=7;k>=0;k--)
        if ((vrsta[j]==3) && (vrsta[k]>=2) && (j!=k))  //ful
        return 500+10*j+k;
      for (j=7;j>=0;j--)
      if (pom[8*i+j]) //najjaca karta boje
        return 400+j; //boja
    }
   for (j=7;j>=0;j--)
     if (vrsta[j]==4)
       return 600+j;  //poker, nije bilo boje
   for (j=7;j>=0;j--)
    for (k=7;k>=0;k--)
      if ((vrsta[j]==3) && (vrsta[k]>=2) && (j!=k))  //ful, nije bilo boje
      return 500+10*j+k;
   for (j=7;j>3;j--)
    if (vrsta[j]&&vrsta[j-1]&&vrsta[j-2]&&vrsta[j-3]&&vrsta[j-4])
      return 300+j; //kenta, j najjaca karta
   for (i=7;i>=0;i--)
     if (vrsta[i]==3)
```

```
      return 200+i; //triling
    for (i=7;i>0;i--)
      if (vrsta[i]==2)
    {
      for (j=i-1;j>=0;j--)
        if (vrsta[j]==2)
         return 100+10*i+j; //dva para
    }
    for (i=7;i>=0;i--)
      if (vrsta[i]==2)
      return 10+i; //par
    return 0; //nista ;-(
}


void izracunaj()
{
   int i,naj=0,max=0,t;
   bool jednako=true;
   for (i=0;i<n;i++)
   {
     t=skor(i);
   if (t>max)
   {
     max=t;
     naj=i;
     jednako=false;
   }
   else if (t==max)
     jednako=true;
   }
  ukupno++;
  if (jednako) {nereseno++; }
  else dobio[naj]++;
}


void gen(int k,int min)
{
  if (k==5)
    izracunaj();
  else
    for (int i=min;i<32;i++)
    if (!karte[i])
    {
       karte[i]=true;
     sto[i]=true;
     gen(k+1,i+1);
     karte[i]=false;
     sto[i]=false;
    }
}


int main()
{
   int i,t;
   string s;
   fin>>n;
   init(karte);
   ukupno=0;
   nereseno=0;
   for (i=0;i<n;i++)
   {
       dobio[i]=0;
       init(igrac[i]);
     fin>>s;
```

```
      t=0;
      switch (s[0])
      {
         case 'T':t+=8;
       case 'H':t+=8;
       case 'K':t+=8;
      }
      fin>>s;
      switch (s[0])
      {
          case 'A':t+=7;break;
        case 'K':t+=6;break;
        case 'Q':t+=5;break;
        case 'J':t+=4;break;
        case '1':t+=3;break;
        default: t+=s[0]-'7';
      }
      igrac[i][t]=true;
      karte[t]=true;
      fin>>s;
      t=0;
      switch (s[0])
      {
         case 'T':t+=8;
       case 'H':t+=8;
       case 'K':t+=8;
      }
      fin>>s;
      switch (s[0])
      {
          case 'A':t+=7;break;
        case 'K':t+=6;break;
        case 'Q':t+=5;break;
        case 'J':t+=4;break;
        case '1':t+=3;break;
        default: t+=s[0]-'7';
      }
      igrac[i][t]=true;
      karte[t]=true;
    }
    fin.close();
    init(sto);
    gen(0,0);
    for (i=0;i<n;i++)
      fout<<dobio[i]*100.0/ukupno<<endl;
    fout<<nereseno*100.0/ukupno<<endl;
    fout.close();
    return 0;
}
```

fajl: poker2.cpp

```
#include <iostream>
#include <string>
#include <map>
#include <limits.h>
using namespace std;

#define FOR(i,j) for(int i=0;i<j;i++)
#define FORi(i,j,k) for(int i=j;i<k;i++)
#define C(i) (i<<3)

int cads[32] = {0},table[32] = {0}, player[10][32] = {0}, won[10] = {0};
int total = 0, draw = 0, N;
```

```
int skor(int brojplayera)
{
    int suit[4] = {0}, kind[8] = {0}, hand[32] = {0}, score = 0;

    FOR(i,4) FOR(j,8) {
        int b = (player[brojplayera][8*i+j]||table[8*i+j]);
        suit[i]+=b;
        kind[j]+=b;
        hand[8*i+j]=b;
    }

    FOR(i,4) {
        score >?= (hand[8*i+7]&&hand[8*i+6]&&hand[8*i+5]&&hand[8*i+4]&&hand[8*i+3])*800;
        if (suit[i]>=5) {
            FORi(j,4,7) score >?= (hand[8*i+j]&&hand[8*i+j-1]&&hand[8*i+j-2]&&hand[8*i+j-
3]&&hand[8*i+j-4]) * (700+j);
            FOR(j,8) score >?= (hand[8*i+j]) * (400+j);
        }
    }

    FOR(j,8) {
        score >?= (kind[j]==4)*(600+j);
        FOR(k,8) score >?= ((j!=k) && (kind[j]==3) && (kind[k]>=2)) * (500+10*j+k);
        score >?= ((j>3)&&(kind[j]&&kind[j-1]&&kind[j-2]&&kind[j-3]&&kind[j-4])) * (300+j);
        score >?= (kind[j]==3)*(200+j);
        FOR(k,8) score >?= ((j!=k) && (kind[j]>=2) && (kind[k]>=2)) * (100+10*j+k);
        score >?= (kind[j]==2) * (10+j);
    }

    return score;
}


void izracunaj()
{
    int naj=0,max=0,t,jednako=0;
    FOR(i,N) {
        t=skor(i);
        jednako |= (t==max);
        if (t>max) {
            max=t;
            naj=i;
            jednako=0;
        }
    }
    total++;
    draw += jednako;
    won[naj] += 1 - jednako;
}


void gen(int k,int min) {
    if (k==5)
        izracunaj();
    else
        FORi(i,min,32)
            if (!cads[i]) {
                cads[i]=table[i]=1;
                gen(k+1,i+1);
                cads[i]=table[i]=0;
            }
}


int main()
{
```

```
    map<char,int> M;
M['A']=7;M['K']=6;M['Q']=5;M['J']=4;M['1']=3;M['9']=2;M['8']=1;M['7']=0;
    map<char,int> Q; Q['T']=24;Q['H']=16;Q['K']=8;Q['P']=0;

    string p,q;
    cin >> N;
    FOR(i,2*N) {
     cin >> q >> p;
     int t = M[p[0]]+Q[q[0]];
     player[i/2][t] = cads[t] = 1;
    }

    gen(0,0);

    FOR(i,N) cout<<won[i]*100.0/total<<endl;
    cout<<draw*100.0/total<<endl;

    cout << total << endl;

    cin >> N;
    return 0;
}
```

fajl: poker.pas

```
const
   fin = 'poker.in';
   fout = 'poker.out';
   MaxN = 10;

var
   n : Integer;
   c : array[1..MaxN, 1..2] of Integer;
   p : array[0..MaxN] of Double;



   procedure Solve;
   var
      t : array[1..5] of Integer;
      t1, t2, t3, t4, t5 : Integer;

      function CalcPower(t6, t7 : Integer) : Integer;
      var
         q : array[1..7] of Integer;
         s : array[1..5] of Integer;
         maxPower : Integer;

         function Evaluate : Integer;
         var
            i, j : Integer;
            k : Integer;

            count : array[0..7] of Integer;
            flush, straight : Boolean;
            c4, c3, c2h, c2l, c1 : Integer;
         begin
            // counting numbers
            for i := 0 to 7 do
               count[i] := 0;
            for i := 1 to 5 do
               inc(count[s[i] mod 8]);

            // flush test
            flush := true;
            for i := 2 to 5 do
               if (s[i] div 8) <> (s[1] div 8) then
```

```
            begin
               flush := false;
               break;
            end;

         // straight test
         straight := true;
         k := 0;
         for i := 0 to 7 do
         begin
            if count[i] = 1 then
               inc(k)
            else
               if (count[i] > 1) or ((k > 0) and (k < 5)) then
               begin
                  straight := false;
                  break;
               end;
         end;

         c4 := -1; c3 := -1; c2h := -1; c2l := -1;
         for i := 0 to 7 do
         begin
            if count[i] = 4 then c4 := i;
            if count[i] = 3 then c3 := i;
            if count[i] = 2 then
            begin
               c2l := c2h;
               c2h := i;
            end;
            if count[i] > 0 then c1 := i;
         end;


         if flush and straight and (c1 = 7) then begin Evaluate := 9 * 64 +   0 * 8 +
0; exit; end;
         if flush and straight              then begin Evaluate := 8 * 64 +  c1 * 8 +
0; exit; end;
         if (c4 <> -1)                      then begin Evaluate := 7 * 64 +  c4 * 8 +
0; exit; end;
         if (c3 <> -1) and (c2h <> -1)      then begin Evaluate := 6 * 64 +  c3 * 8 +
c2h; exit; end;
         if flush                           then begin Evaluate := 5 * 64 +  c1 * 8 +
0; exit; end;
         if straight                        then begin Evaluate := 4 * 64 +  c1 * 8 +
0; exit; end;
         if (c3 <> -1)                      then begin Evaluate := 3 * 64 +  c3 * 8 +
0; exit; end;
         if (c2h <> -1) and (c2l <> -1)     then begin Evaluate := 2 * 64 + c2h * 8 +
c2l; exit; end;
         if (c2h <> -1)                     then begin Evaluate := 1 * 64 + c2h * 8 +
0; exit; end;
            Evaluate := 0;
      end;

      procedure EvaluateSubset(p, k : Integer);
      var
        j, t : Integer;
      begin
         if k = 5 then
         begin
            t := Evaluate;
            if t > maxPower then
               maxPower := t;

            exit;
         end;
```

```
         for j := p to 3 + k do
         begin
            s[k+1] := q[j];
            EvaluateSubset(j + 1, k + 1);
         end;
      end;

   var
      i : Integer;

   begin
      q[1] := t1; q[2] := t2; q[3] := t3; q[4] := t4; q[5] := t5;
      q[6] := t6;
      q[7] := t7;

      maxPower := 0;
      EvaluateSubset(1, 0);
      CalcPower := maxPower;
  end;

var
   used : array[0..31] of Boolean;
   i, h : Integer;
   power : array[1..MaxN] of Integer;
   wins : array[0..MaxN] of Longint;
   total : Longint;
   maxPower : Integer;
   bestPlayer : Integer;
begin
   for i := 0 to 31 do
      used[i] := false;

   for i := 1 to n do
   begin
      if c[i, 2] < c[i, 1] then
      begin
         h := c[i, 1];
         c[i, 1] := c[i, 2];
         c[i, 2] := h;
      end;

      used[c[i, 1]] := true;
      used[c[i, 2]] := true;
      wins[i] := 0;
   end;
   total := 0;

   for t1 := 0 to 31 - 4 do
   if not used[t1] then
   begin
      used[t1] := true;

      for t2 := t1 + 1 to 31 - 3 do
      if not used[t2] then
      begin
         used[t2] := true;

         for t3 := t2 + 1 to 31 - 2 do
         if not used[t3] then
         begin
            used[t3] := true;

            for t4 := t3 + 1 to 31 - 1 do
            if not used[t4] then
            begin
               used[t4] := true;
```

```
               for t5 := t4 + 1 to 31 do
               if not used[t5] then
               begin
                  for i := 1 to n do
                     power[i] := calcPower(c[i, 1], c[i, 2]);

                  maxPower := -1;
                  for i := 1 to n do
                  begin
                     if power[i] > maxPower then
                     begin
                        maxPower := power[i];
                        bestPlayer := i;
                     end else
                        if power[i] = maxPower then
                           bestPlayer := 0;
                  end;

                  inc(total);
                  inc(wins[bestPlayer]);
               end;

               used[t4] := false;
            end;

            used[t3] := false
         end;

         used[t2] := false;
      end;

      used[t1] := false;
   end;

   for i := 0 to n do
      p[i] := 100 * wins[i] / total;
end;



procedure ReadInput;
var
   f : Text;
   i : Integer;

   function ReadCard : Integer;
   var
      ch : Char;
      suit, number : Integer;
   begin
      Read(f, ch);
      case ch of
         'P' : suit := 0;
         'K' : suit := 1;
         'H' : suit := 2;
         'T' : suit := 3;
      end;
      Read(f, ch);
      Read(f, ch);
      case ch of
         'A' : number := 7;
         'K' : number := 6;
         'Q' : number := 5;
         'J' : number := 4;
         '1' : begin number := 3; Read(f, ch); end;
         '9' : number := 2;
```

```
            '8' : number := 1;
            '7' : number := 0;
         end;
         Readln(f);
         ReadCard := suit * 8 + number;
      end;

   begin
      Assign(f, fin);
      Reset(f);
      Readln(f, n);
      for i := 1 to n do
      begin
         c[i][1] := ReadCard;
         c[i][2] := ReadCard;
      end;
      Close(f);
   end;


   procedure WriteOutput;
   var
      f : Text;
      i : Integer;
   begin
      Assign(f, fout);
      Rewrite(f);
      for i := 1 to n do
         Writeln(f, p[i]:0:2);
      Writeln(f, p[0]:0:2);
      Close(f);
   end;



begin
   ReadInput;
   Solve;
   WriteOutput;
end
```