ispitni centar
PRAVA
MJERA
ZNANJA

DRŽAVNO TAKMIČENJE 2015.

ŠIFRA UČENIKA

SREDNJA ŠKOLA

PROGRAMIRANJE

UKUPAN BROJ OSVOJENIH BODOVA

Test pregledala/pregledao

.....

.....

Podgorica, ... 20..... godine

Uputstva takmičarima

Ovo takmičenje sastoji se od rješavanja **3 problemska zadatka** u vremenu od **4 sata** (240 minuta). Zadatke je potrebno rješavati u jednom od sljedećih programskih jezika: Pascal, C, C++ ili Java. Takmičari koji koriste Pascal moraju programirati u programskom alatu **FreePascal ili TurboPascal.** Takmičari u C-u i C++-u moraju koristiti programske alate **CodeBlocks**, **DJGPP, DevCpp ili GCC**. Za programski jezik Java predviđena je upotreba platforme **Eclipse**. Dozvoljeno je koristiti editor po izboru i pomoću navedenih alata prevoditi izvorni kod u izvršnu datoteku.

Tokom takmičenja **ne smijete komunicirati** ni sa jednom osobom, osim dežurne osobe takmičenja. To znači da morate **raditi samostalno** i **ne smijete koristiti Internet**. Takođe, zabranjena je upotreba bilo kakvih ranije napisanih programa ili dijelova programa.

Po isteku vremena predviđenog za takmičenje, na desktopu u folderu sa imenom ***Takmicenje2015*** moraju se nalaziti datoteke sa snimljenim izvornim kôdovima rješenja. Nakon takmičenja, komisija će testirati vaša rješenja na ranije izabranim test podacima i dodijeliti vam određeni broj bodova. Na kraju svakog zadatka dati su primjeri test podataka. Ti primjeri služe da bi vam tekst zadataka bio što je moguće jasniji te za provjeru formata ulaza i izlaza, a ne služe za provjeru ispravnosti vašeg programa. Ako vaš program radi na tim primjerima, to **nije garancija** da će raditi na službenim podacima za testiranje.

Zadaci ne nose jednak broj bodova. Lakše i brže rješivi zadaci nose manje bodova, dok teži nose više bodova. Svaki test podatak u nekom zadatku nosi jednak broj bodova. Ukupan broj bodova na nekom zadatku jednak je zbiru bodova test podataka koji se poklapaju sa službenim rješenjem. Ukupan broj bodova jednak je zbiru bodova na svim zadacima.

Sve informacije o zadacima (ime zadatka, vremensko i memorijsko ograničenje, način bodovanja) možete naći na uvodnoj stranici s naslovom *Zadaci*. Ako vam nije jasno nešto u vezi načina organizacije ovog takmičenja, odmah postavite pitanje dežurnom da vam to razjasni.

Tokom cijelog takmičenja možete postavljati pitanja dežurnom u vezi zadataka. Dozvoljena su pitanja **koja razjašnjavaju nejasnoće u tekstu zadatka**. Ne smijete postavljati pitanja u vezi rješavanja zadataka. Prije nego postavite pitanje, pročitajte još jednom zadatak, jer je moguće da ste u prethodnom čitanju preskočili dio teksta zadatka.

VAŽNO za C/C++!

Glavni program (glavna funkcija) **mora** biti deklarisan kao: `int main(void) { … }`.

Program mora završiti svoje izvođenje naredbom `return 0;` unutar funkcije `main` ili naredbom `exit(0);`.

Zabranjeno je koristiti biblioteke <conio.h> i <cconio>, kao i sve funkcije deklarisane u ovim bibliotekama (npr. clrscr(); getch(); getche(); i sl.). Zabranjeno je koristiti i **sve** sistemske (nestandardne) biblioteke.

Zabranjeno je koristiti funkcije itoa() i ltoa() jer one ne postoje u standardu jezika C/C++. Umjesto tih funkcija možete koristiti funkciju sprintf() deklarisanu u <stdio.h> i <cstdio>, koja ima i veće mogućnosti primjene,

Dozvoljeno je koristiti sve ostale standardne biblioteke (koje su dio jezika), uključujući i STL (Standard Template Library) u jeziku C++.

VAŽNO za Pascal!

Program **mora** regularno završiti svoje izvođenje naredbom end. unutar glavnog programa ili naredbom halt;.

Zabranjeno je koristiti bilo kakve biblioteke, a posebno biblioteku crt, tj. zabranjeno je u programu imati direktivu uses. To znači da u programu ne smije biti naredbi clrscr() i readkey().

Nepoštovanje ovih pravila ili nepridržavanje formata izlaznih podataka rezultiraće nepovratnim gubitkom bodova. Nemojte štampati ništa što se u zadatku ne traži, kao npr. poruke tipa 'Rjesenje je:' ili 'Unesite brojeve' i slično!

Srećno i uspješno takmičenje!

Zadaci

Zadatak	Zadatak1	Zadatak2	Zadatak3
Izvorni kôd	zadatak1.java zadatak1.pas zadatak1.c zadatak1.cpp	zadatak2.java zadatak2.pas zadatak2.c zadatak2.cpp	zadatak3.java zadatak3.pas zadatak3.c zadatak3.cpp
Memorijsko ograničenje	64 MB	256 MB	64 MB
Vremensko ograničenje (po test podatku)	1 sekunda	2 sekunde	2 sekunde
Broj test podataka	10	10	10
Broj bodova (po test podatku)	3	3.5	3.5
Ukupno bodova	**30**	**35**	**35**

Napomena: Program u C-u i C++-u treba kompajlirati sa sljedećim opcijama: *–O2 –lm –static*, a program u Pascalu sa *–O1 –XS*.

Zadatak 1 – Analiza glasanja

Na predsjedničkim izborima učestvuje n kandidata. Da bi spriječila zloupotrebe, izborna komisija je naručila mašinu za analizu glasačkih listića. Listić sadrži po jedno kvadratno polje za svakog predsjedničkog kandidata. Svaki glasač može označiti tačno jednog kandidata. Ako nijedno polje nije označeno ili su označena dva ili više polja, listić je nevažeći. Listići se skeniraju i kreira se specijalni niz karaktera dužine n: 'X' predstavlja označeno polje, a '.' predstavlja neoznačeno polje.
Sada je potrebno napisati program koji na osnovu dobijenih nizova karaktera svih glasača generiše izvještaj o izborima. Predsjedničke kandidate treba poređati u opadajući poredak po broju glasova. Ako dva kandidata imaju isti broj glasova, poređati ih kako su poređani na glasačkom listiću. Za svakog kandidata izračunati i štampati procenat osvojenih glasova. Posljednji red izvještaja sadrži procenat nevažećih listića.

Ulazni podaci

Prvi red sadrži dva cijela broja n i m – broj kandidata i broj glasačkih listića ($2 \le n \le 10$; $1 \le m \le 1000$). Sljedećih n redova sadrže imena kandidata, po jedno u redu, gdje je svako ime niz od najviše 100 slova engleske abecede. Ne postoji kandidat sa imenom "Invalid". Sljedećih m redova sadrže opise glasačkih listića – niz karaktera koji se sastoji samo od simbola 'X' i '.'.

Izlazni podaci

Štampati n+1 red. Prvih n redova su rezultati kandidata izraženi procentima, po jedan kandidat u redu. Za svakog kandidata štampati njegovo ili njeno ime, jedan blanko, rezultat u procentima zaokružen na dvije decimale i simbol '%'. Posljednji red sadrži procenat nevažećih listića: riječ "Invalid", jedan blanko, rezultat u procentima zaokružen na dvije decimale i simbol '%'. Ako je broj na sredini između dva broja, štampati veći od njih (npr. štampati "12.35" ako je broj 12.345).

Test primjeri

Ulaz	**Izlaz**
4 7 Larisa Anisa Darko Milan .X.. X... .... ..X. ..XX ..X. ..X.	Darko 42.86% Larisa 14.29% Anisa 14.29% Milan 0.00% Invalid 28.57%

Rješenje: Ovaj zadatak je klasičan primjer simulacije – samo je potrebno pažljivo implementirati opisani postupak.

C++

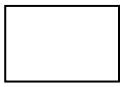
```
#include <iostream>
#include <stdio.h>
#include <string.h>
using namespace std;

int main()
{
    int brKand = 0, brGlasova = 0, brNevazi = 0;
    scanf("%d%d%*c", &brKand, &brGlasova);
    char kandidati[11][101];
    int kandidatiGlas[11] = {0};

    for(int i = 0; i < brKand; i++)
    {
        gets(kandidati[i]);
    }
    for(int i = 0; i < brGlasova; i++)
    {
        char linija[11];
        gets(linija);
        int len=strlen(linija);
        int brojX = 0;
        int kand = -1;
        for(int j = 0;  j < len; j++)
            if (linija[j] == 'X')
            {
                brojX++;
                kand = j;
            }

        if (brojX != 1)
            brNevazi++;
        else
            kandidatiGlas[kand]++;
    }
    for(int i = 0; i < brKand; i++)
    {
        int max = i;
        for(int j = 0; j < brKand ; j++)
        {
            if (kandidatiGlas[j] > kandidatiGlas[max])
                max = j;
        }
        printf("%s %.2f%%\n", kandidati[max], 100.0f *
(kandidatiGlas[max]) / (float)brGlasova);
        kandidatiGlas[max] = -1;
    }
    printf("%s %.2f%%", "Invalid", 100.0f * (brNevazi) /
(float)brGlasova);
    return 0;
}
```

Zadatak 2 – Simpatični brojevi



Prirodan broj A je *simpatičniji* od prirodnog broja B ako je broj cifara potrebnih za dekadni zapis broja A manji od broja cifara potrebnih za dekadni zapis broja B. Na primjer, 55 je simpatičniji od 12, a 12 je simpatičniji od 123.
Za dati prirodan broj N odrediti najveći cio broj X takav da je X < N i X je simpatičniji od N.
Ulaz: Prvi i jedini red ulaza sadrži cio broj N, $1 \le N \le 2^{31} - 1$.
Izlaz: Štampati cio broj X. Ako ne postoji cio broj simpatičniji od N, štampati 0.
Primjer:

Ulaz	**Ulaz**
111	765437654
Izlaz	**Izlaz**
0	765377777

Rješenje: Očigledno rješenje je da smanjujemo broj N sve dok ne naiđemo na prvi simpatičniji broj. Ovo je rješenje složenosti $O(N \times log\ N)$ i presporo je da se dobiju svi bodovi.
Posmatrajmo dekadne zapise brojeva $N = n_1\ n_2 \ldots\ n_K$ i $X = x_1\ x_2 \ldots\ x_M$. Kako je $X < N$, moguća su dva slučaja:

- $M < K$,
- $M = K$ i postoji p takvo da je $x_1 = n_1, x_2 = n_2, \ldots, x_{p-1} = n_{p-1}$ i $x_p < n_p$

U slučaju kada je $M < K$, lako se vidi da je $M = K - 1$ i $x_1 = x_2 = \ldots = x_{K-1} = 9$.
U drugom slučaju sve cifre poslije pozicije p moraju biti jednake (tj. $x_{p+1} = x_{p+2} = \ldots = x_K$). Zaista, ako bi neke bile različite, možemo ih sve zamijeniti najvećom od njih i dobiti novi broj $X' > X$, pri čemu je X' simpatičan bar koliko X i $X' < N$. Otuda, sve moguće vrijednosti za X dobijamo pregledom svih kombinacija $p = 1 \ldots K$, $x_p = 0 \ldots n_p - 1$, $x_{p+1} = 0 \ldots 9$. Složenost rješenja je $O(10^2 \times log^2 N)$..

C++

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

std::vector<int> u, bestAns(1), digs, w(10);
int cmpl;

int value(const std::vector<int>& d)
{
  int res = 0;
  for (int i = d.size(); i > 0; --i) res = (res * 10) + d[i - 1];
  return res;
}

void checkSol(void)
```

```
{
  std::vector<int> ans(20);
  int ml;
  int dl = digs.size();
  int j;

  if (!u.size()) return;

  for (ml = digs.size(); ml > 0 && w[digs[ml - 1]]; --ml)
    if (digs[ml - 1] > u[0]) dl = ml;

  if (ml == 0 || digs[ml - 1] < u[0]) ml = dl;

  j = 0;
  while (j + 1 < u.size() && u[j + 1] < digs[ml - 1]) j++;
  ans[ml - 1] = u[j] < digs[ml - 1] ? u[j] : 0;

  for (int i = digs.size(); i > ml; --i) ans[i - 1] = digs[i - 1];
  for (int i = ml - 1; i > 0; --i) ans[i - 1] = u[u.size() - 1];

  if (value(ans) > value(bestAns)) bestAns = ans;

  return;
}

void rec(int l)
{
  if (u.size() >= cmpl) return;

  checkSol();

  for (int i = l; i < 10; ++i)
  {
    u.push_back(i); w[i] = 1;
    rec(i + 1);
    u.pop_back(); w[i] = 0;
  }
}
int main(int argc, char** argv)
{
  int n, i;
  std::fstream input("input.txt", std::ios_base::in);
  std::fstream output("output.txt", std::ios_base::out);

  input >> n;
  for (int m = n; m > 0; m /= 10) digs.push_back(m % 10);

  for (i = 0; i < digs.size(); ++i)
  {
    bool frst = true;
    for (int j = 0; j < i; ++j) frst &= digs[i] != digs[j];
    cmpl += frst ? 1 : 0;
  }

  rec(0);
  for (i = bestAns.size(); i > 1 && bestAns[i - 1] == 0; --i);
  for (; i > 0; --i) output << bestAns[i - 1];

  return 0;
}
```

Zadatak 3 – Brod

Petru je dosadio posao programera, pa se odselio u jednu afričku zemlju i tamo pokrenuo posao prevoženja putnika brodom. Petar prevozi putnike po rijeci uz koju se nalaze sela označena brojevima od 0 do M, redom. Razmak između dva susjedna sela iznosi tačno 1 kilometar. Petar živi u selu broj 0 i svakog dana putuje do sela M i prevozi usput putnike. Danas je broj putnika jednak N a za svakog od njih je poznato polazište i odredište.

Petrov brod je dovoljno veliki i može da primi veliki broj putnika. Npr. ako osoba A putuje iz sela 2 u selo 8, a osoba B iz sela 6 u selo 4, tada Petar, krenuvši iz sela 0, može napraviti sljedeće: pokupiti osobu A u selu 2, potom pokupiti osobu B u selu 6, vratiti se do sela 4, tamo pustiti osobu B, voziti do sela 8 i tamo iskrcati osobu A i produžiti dalje do sela M. (Ovaj scenario odgovara prvom test primjeru).

Napiši program koji računa ukupan minimalan broj kilometara koji Petar mora preći kako bi, krenuvši iz sela 0, odvezao sve putnike na njihova odredišta, i na kraju stigao u selo M.

Ulazni podaci

U prvom redu ulaza nalaze se prirodni brojevi N i M ($N \leq 300\,000$, $3 \leq M \leq 10^9$) iz teksta zadatka, odvojeni razmakom. U sljedećih N redova nalaze se podaci o putnicima: po dva različita cijela broja veća od 0 i manja od M, koja redom označavaju polazište i odredište dotičnog putnika.

Izlazni podaci

Traženi minimalan broj kilometara puta.

Bodovanje

U test podacima ukupno vrijednim 30% bodova, broj N će biti manji ili jednak 5000.

Test primjeri

Ulaz	Izlaz
2 10 2 8 6 4	14
8 15 1 12 3 1 3 9 4 2 7 13 12 11 14 11 14 13	27

Rješenje:

Očigledno je da treba naći put sa što je moguće manje vraćanja. Ako dođemo do sela b pa se vratimo do sela a, i ponovo nastavimo ka selu b, tada smo interval [a.b] prešli tri puta. Ako nema vraćanja, tad bi taj interval prešli samo jednom. Jasno je da svaki interval po kojem se vraćamo prelazimo dva puta, pa je konačno rješenje jednako:

M + 2*(ukupna dužina intervala po kojima se vraćamo).

Da bi ovaj put bio najkraći mogući, potrebno je minimizovati dužinu intervala vraćanja. Osobe koje putuju slijeva udesno (tj. od sela a do sela b gdje je a<b) možemo potpuno zanemariti, jer ćemo njih prevesti u svakom slučaju. Osobe koje putuju zdesna na lijevo (tj. od sela a do sela b gdje je a>b) su važne jer svaka od njih definiše jedan interval po kojem se vraćamo. Moguće je da je taj interval dio nekog većeg intervala. Sada odredimo uniju svih intervala po kojima se moramo vraćati i dužina te unije biće traženo dužina.
Ovo se može uraditi primjenom algoritma "sweep line": sortiramo početke i krajeve intervala u ratućem poretku i idući slijeva udesno označavamo događaje tipa tipa „početak intervala“ i „kraj intervala“. Složenost algoritma je O(NlogN).

C++

```
#include<cstdio>
#include<vector>
#include<algorithm>
using namespace std;

vector< pair<int,int> > veze;

int main()
{
    int n, m, unija = 0;
    scanf("%d%d", &n, &m);
    for(int i=0; i<n; i++)
    {
        int x, y; scanf("%d%d", &x, &y);
        if( x < y ) continue;
        veze.push_back( make_pair(y,x) );
    }
    sort(veze.begin(), veze.end());
    int odakle=0, dokle=0, intervala=0;
    for(int i=0; i<(int)veze.size(); i++)
    {
        if( veze[i].first > dokle )
        {
            unija += dokle-odakle;
            odakle = veze[i].first;
            dokle = veze[i].second;
            intervala++;
```

```
            }
            else dokle = max(dokle, veze[i].second);
        }
        unija += dokle-odakle;
        printf("%lld\n", (long long)m + 2*unija);
        return 0;
}
```