ispitni centar
PRAVA
MJERA
ZNANJA

DRŽAVNO TAKMIČENJE 2018.

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 ***Takmicenje2018*** 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 – Nove riječi

Ana je otkrila novi metod stvaranja riječi od date dvije riječi. Uzme bilo koji neprazni prefiks prve riječi i bilo koji neprazni sufiks druge riječi I nadoveže sufiks na prefiks. Na primjer, na taj način od riječi "tree" i "heap", može dobiti riječi "treap", "tap" ili "theap".
Ana je izabrala dvije riječi i sada je zanima koliko različitih riječi može dobiti primjenom opisanog pravila.

Ulazni podaci

Prva dva reda ulaza sadrže dvije rječi koje je Ana odabrala. Dužina riječi je između 1 i 100000 karaktera i koriste se samo mala slova engleske abecede.

Izlazni podaci

Štampati samo jedan cio broj – koliko se različitih riječi može stvoriti od dvije date riječi.

Test primjeri

Ulaz	Izlaz
cat dog	9
tree heap	14

Rješenje:

Rješenje za oko 50% bodova:

Generisati sve moguće riječi, smještati ih u skup (npr. <set> u jeziku C++, <HashSet> ili <TreeSet> u jeziku Java) i štampati broj elemenata skupa.

Rješenje za 100% bodova:

Pažljivo provjeriti kad se pojavljuju duplikati riječi. Na primjer, od riječi tree i heap moguće je napraviti riječ treap na više načina. Neka su dati stringovi S1 i S2. Označimo sa A(x) koliko se puta slovo x pojavljuje u S1 bez prvog slova i sa B(x) koliko se puta pojavljuje slovo x bez posljednjeg slova. Tada je rješenje dato formulom: $length(S1) * length(S2) - \sum_{x='a'}^{'z'} A(x)B(x)$

C++ kod:

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

```
using namespace std;

int main() {
  string a, b;
  getline(cin, a);
  getline(cin, b);

  vector<int> count[2];
  count[0].resize(26);
  count[1].resize(26);

  for (int i = 1; i < a.size(); i++) {
    count[0][a[i]-'a']++;
  }
  for (int i = 0; i < b.size() - 1; i++) {
    count[1][b[i]-'a']++;
  }

  long long ans = a.size() * 1LL * b.size();
  for (int i = 0; i < 26; i++)
    ans -= count[0][i] * 1LL * count[1][i];
  cout << ans << endl;
  return 0;
}
```

Zadatak 2 – Projekat

Ana je na svom projektu iz elektronike koristila LED ekrane. Svaki od ekrana ima sedam segmenata, a potrošnja energije je proporcionalna broju osvijetljenih segmenata (npr. broj 9 troši dva puta više energije od broja 7).

Ana se zapitala koliki je maksimalni mogući zbir cifara na ekranima ako ima bateriju koja može napajati tačno n segmenata i želi da svih n segmenata na ekranima budu osvijetljeni.

Ulazni podaci

Jedini red ulaza sadrži cio broj **n** – broj segmenata koje treba osvijetliti ($2 \leq \mathbf{n} \leq 10^6$).

Izlazni podaci

Jedini red izlaza sadrži jedan cio broj – najveći mogući zbir cifara koje mogu biti prikazane istovremeno.

Test primjeri

Ulaz	Izlaz	Objašnjenje
4	4	Jedna cifra 4 će biti prikazana
7	11	Cifre 4 i 7 će biti prikazane
6	14	Dvije cifre 7 će biti prikazane

Rješenje:

Prvo rješenje: Dinamičko programiranje: $n_i = \underbrace{max(d + n_{i-s_d})}_{d=0..9}$, gdje je s_d broj segmenata za cifru d.

Drugo rješenje: Pažljivo pregledamo slučajeve n mod 3:

```
n mod 3 == 0: 7n/3
n mod 3 == 1: 7(n - 4)/3 + 4
n mod 3 == 2: 7(n - 2)/3 + 1
```

C++ kod:

```
#include <cstdio>
using namespace std;
int main() {

  int n;
```

```
  scanf("%d", &n);
  if (n % 3 == 2) {
    printf("%d\n", n / 3 * 7 + 1);
  } else if (n % 3 == 1) {
    printf("%d\n", n / 3 * 7 - 3);
  } else {
    printf("%d\n", n / 3 * 7);
  }
  return 0;
}
```

Zadatak 3 – Zgrade

Ana kupuje rođendanski poklon najboljem drugu i može birati između n predmeta numerisanih brojevima od 1 do n. Ona želi da kupi robota, koji je predmet broj 1, ali postoje različiti mehanizmi kako može nabavljati predmete:

- može kupiti predmet; cijena i-tog predmeta je c_i
- može sastaviti predmet od druga dva predmeta. Prodavnica nudi m načina sastavljanja, tako što je moguće uzeti dva različita predmeta i od njih napraviti jedan.

Pomozite Ani da potroši najmanje novca za kupovinu robota (predmeta broj 1).

Ulazni podaci

Prvi red ulaza sadrži cijele brojeve n i m ($1 \le n \le 10\,000$; $0 \le m \le 100\,000$) – broj različitih predmeta i broj načina sastavljanja.
Drugi red sadrži n cijelih brojeva c_i – vrijednosti predmeta ($0 \le c_i \le 10^9$).
Sljedećih m redova opisuje načine sastavljanja. Svaki red sadrži po tri različita cijela broja a_i, x_i, y_i, gdje je a_i predmet koji se može sastaviti od predmeta x_i i y_i ($1 \le a_i; x_i; y_i \le n$; $a_i \ne x_i$; $x_i \ne y_i$; $y_i \ne a_i$).

Izlazni podaci

Štampati jedan cio broj – kolika je najmanja moguća količina novca koji Ana može potrošiti.

Test primjeri

Ulaz	Izlaz
5 3 5 0 1 2 5 5 2 3 4 2 3 1 4 5	2

Pojašnjenje test primjera

Cijena predmeta 1 je 5. Možemo ga sastaviti od predmeta 4 i 5 po cijeni 2+5=7. Međutim, ako prvo sastavimo predmet 4 od predmeta 2 i 3 (po cijeni 0+1=1) a zatim predmet 5 od predmeta 2 i 3 (po cijeni 0+1=1), tada se predmet 1 može dobiti po cijeni 1+1=2.

Rješenje:

Kreirati graf koji ima n+1 čvor: jedan početni čvor (koji se obično naziva source) a ostalih n čvorova su dati predmeti. Povezati source sa svim čvorovima tako da je cijena grane (source , i) bude c_i. Ako se predmet x može sastaviti od predmeta y i z, tada se dodaje grana (y,x) sa cijenom c[z] i grana (z,x) sa cijenom c[y]. U takvom grafu se za svaki čvor p odredi dužina najkraćeg puta od source do p.

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

using namespace std;

const int N = 20010;

int cost[N];
bool visited[N];
vector < pair <int, int> > g[N];

int main() {
  int n, m;
  scanf("%d %d", &n, &m);
  for (int i = 1; i <= n; i++) scanf("%d", cost + i);
  for (int i = 0; i < m; i++) {
    int a, b, c;
    scanf("%d %d %d", &c, &a, &b);
    g[a].push_back(make_pair(b, c));
    g[b].push_back(make_pair(a, c));
  }
  for (int i = 1; i <= n; i++) visited[i] = false;
  for (int it = 0; it < n; it++) {
    int mn = (int)2e9, km = -1;
    for (int i = 1; i <= n; i++)
      if (!visited[i] && cost[i] < mn) {
        mn = cost[i];
        km = i;
      }
    visited[km] = true;
    int sz = g[km].size();
    for (int j = 0; j < sz; j++) {
      int tot = mn + cost[g[km][j].first];
      if (tot < cost[g[km][j].second]) {
        cost[g[km][j].second] = tot;
      }
    }
  }
  printf("%d\n", cost[1]);
  return 0;
}
```