ispitni centar
PRAVA
MJERA
ZNANJA

DRŽAVNO TAKMIČENJE 2014.

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 ***Takmicenje2014*** 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 – Homeopatija

Baštovan Marko praktikuje više vrsta alternativne medicine, uključujući i homeopatiju. On vjeruje da ako rastvorite određene supstance u vodi ili alkoholu i dovoljno jako protresete, sigurno dobijete lijek za mnoge bolesti. Istu praksu Marko primjenjuje i na biljkama u svojoj bašti. Međutim, Marko je i veliki obožavalac matematike, pa zahtijeva da koncentracija supstance koju rastvara u vodi mora biti recipročna prirodom broju, tj. biti oblika 1/n, gdje je n prirodan broj. Markova komšinica Lejla, uvidjevši da ova „biljna homeopatija" uspijeva, zatražila je od Marka „čarobni biljni lijek". Marko ima još jednu kantu sa lijekom koncentracije 1/n. Naravno, ne želi da u potpunosti ostane bez svog magičnog lijeka, pa hoće da ga podijeli tako da i on i Lejla dobiju dio koji je takođe recipročan prirodnom broju. Pomozite Marku da odredi na koliko načina može podijeliti lijek.

Ulazni podaci

Jedini red ulaza sadrži izraz oblika "1/n", bez navodnika, koji predstavlja koncentraciju lijeka u kanti. Garantuje se da je $1 \leq n \leq 10^9$. U ovom redu nema blankova.

Izlazni podaci

U jedini red izlaza štampati broj parova prirodnih brojeva (x,y) koji zadovoljavaju uslove zadatka. Parovi koji se razlikuju samo u poretku brojeva nijesu različiti.

Test primjeri

Ulaz	Izlaz
1/2	1
1/4	3
1/1	1
1/5000	32

Objašnjenje: U prvom primjeru jedini par je (4,4), jer je 1/2 = 1/4+1/4. U drugom primjeru, postoje tri para: (5,20), (6,12) i (8,8).

Rješenje: Treba odrediti broj rješenja jednačine $1/x + 1/y = 1/n$ u skupu prirodnih brojeva. Možemo pretpostaviti da je $x \leq y$, pa dobijamo da je $1/n = 1/x + 1/y \leq 2/x$, pa otuda važi $x \leq 2n$. Sada probamo sve brojeve od n+1 do 2n i prebrojimo koliko njih zadovoljava uslov.

C++ / Java funkcija

```
int solve(int n) {
      int c = 0;
```

```
        for (int x = n+1; x <= n*2; ++x) {
              if (x * n % (x-n) == 0)
                    ++c;
        }
        return c;
}
```

Zadatak 2 – Šifra

Jelena i Nikola se dopisuju, ali ne žele da drugi čitaju njihove poruke. Osim što koriste engleski jezik, oni su dodatno osigurali svoje poruke tako što su uveli svoj sistem kodiranja poruka. Istini za volju, sistem nije baš nov, jer se zasniva na Ohaverovom šifriranju, a ono se bazira na Morzeovoj azbuci, koja kodira svako slovo nizom tačaka i crtica. Sljedeća tabela prikazuje Morzeov kod za sva slova engleske abecede:

A	.-	H	....	O	---	V	...-
B	-...	I	..	P	.--.	W	.--
C	-.-.	J	.---	Q	--.-	X	-..-
D	-..	K	-.-	R	.-.	Y	-.--
E	.	L	.-..	S	...	Z	--..
F	..-.	M	--	T	-		
G	--.	N	-.	U	..-		

Postoje 4 kombinacije tačaka i crtica koje nijesu dodijeljene, pa su ih Jelena i Nikola iskoristili na sljedeći način (ove 4 kombinacije nijesu dio startne Morzeove azbuke):

Donja crta ("_") ..--	Zapeta (",") .-.-
Tačka (".") ---.	Upitnik ("?") ----

U praksi, znaci poruke razdvojeni su kratkom pauzom, koja se obično predstavlja blanko znakom. Npr. poruka NY_REGION će biti šifrirana kao:

```
-. -.-- ..-- .-. . --. .. --- -.
```

Jelena i Nikola kodiraju poruku tako što uklone pauze između slova. Jasno je da tako dobijeni niz znaka može predstavljati više riječi. Na primjer, poruka ".--.-.--" može se tumačiti kao "ACM" ili "ANY", a to nijesu jedine mogućnosti. Ako na kraj stringa dodamo informaciju o dužini, kao npr. ".--.-.-- 242", tada je kod nedvosmislen. Dakle, njihov postupak može se opisati sa sljedeća tri koraka:

1. Poruku konvertujemo Morzeovom azbukom bez pauza, ali dodamo na kraj brojeve koji označavaju dužine Morzeovih kodova.

2. Okrenemo brojeve sa kraja stringa.
3. Konvertujemo tačke i crte nazad u slova koristeći obrnuti niz brojeva kao dužine kodova.

Na primjer, neka šifriramo poruku "AKADTOF_IBOETATUK_IJN". Konvertovanje u Morzeov kod sa dužinama na kraju daje:

.--.-.--..----..-...--..-...---.-.--..--.-..--...----. 232313442431121334242

Okretanjem niza brojeva dobijamo 242433121134244313232, pa sada taj niz koristimo za dešifrovanje i dobijamo originalnu poruku "ACM_GREATER_NY_REGION".
Jasno je da sigurnost ovog metoda nije pretjerano velika, ali Jelena i Nikola misle da je dovoljna za njihove potrebe. Pomozite im da implementiraju ovaj algoritam dešifriranja.

Ulazni podaci

Jedini red ulaza sadrži jednu poruku šifriranu Jeleninim i Nikolinim sistemom. Svaka poruka se sastoji od velikih slova engleske abecede, tačaka, donjih crta, zapeta i upitnika i nije duža od 1000 znaka.

Izlazni podaci

U jedinom redu izlaza štampati dešifriranu poruku.

Test primjeri

Ulaz	Izlaz
FENDSVTSLHW.EDATS,EULAY	FALSE_SENSE_OF_SECURITY
TRDNWPLOEF	CTU_PRAGUE
NTTTGAZEJUIIGDUZEHKUE	TWO_THOUSAND_THIRTEEN
QEWOISE.EIVCAEFNRXTBELYTGD.	QUOTH_THE_RAVEN,_NEVERMORE.
?EJHUT.TSMYGW?EJHOT	TO_BE_OR_NOT_TO_BE?
DSU.XFNCJEVE.OE_UJDXNO_YHU?VIDWDHPDJIKXZT?E	THE_QUICK_BROWN_FOX_JUMPS_OVER_THE_LAZY_DOG

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

```
#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace std;

char* codes[] = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.",
                 "....", "..", ".---", "-.-", ".-..", "--", "-.",
                 "---", ".--.", "--.-", ".-.", "...", "-", "..-",
                 "...-", ".--", "-..-", "-.--", "--.."};
```

```
char* getKod(char c)
{
    if (c >= 'A' && c <= 'Z')
    {
        int num = c - 'A';
        return codes[num];
    }
    if (c=='_') return "..--";
    if (c=='.') return "---.";
    if (c==',') return ".-.-";
    if (c=='?') return "----";
}

char getZnak(char* code)
{
    for(int i = 0; i <= 25; i++)
        if (strcmp(code, codes[i]) == 0)
            return 'A' + i;

    if (strcmp("..--", code)==0) return '_';
    if (strcmp("---.", code)==0) return '.';
    if (strcmp(".-.-", code)==0) return ',';
    if (strcmp("----", code)==0) return '?';
}

int main()
{
    //freopen("in.txt", "r", stdin);
    char linija[1024];
    char outputlinija[1024];
    char sifra[4096];
    int duzine[1024];

    while(!feof(stdin))
    {
        gets(linija);

        int i = 0;
        sifra[0] = '\0';//prazan string
        int duzina = strlen(linija);
        for(i = 0; i < duzina; i++)
        {
            char* tmpKod = getKod(linija[i]);
            strcat(sifra, tmpKod);
            duzine[i] = strlen(tmpKod);
        }

        i = duzina - 1;
        int tekuci = 0;
        int out = 0;
        char kod[10];

        while(i >= 0)
        {
            strncpy(kod, sifra + tekuci, duzine[i]);
            kod[duzine[i]] = '\0';
            tekuci += duzine[i];
            outputlinija[out++] = getZnak(kod);
            i--;
        }
```

```
        outputlinija[out] = '\0';
        printf("%s\n", outputlinija);
    }
    return 0;
}
```

Zadatak 3 – Igra

Dino i Ana igraju jednu zanimljivu igru. Dino na (jako dugačkom) papiru napiše N brojeva. Ana tada uzima olovku i precrta neke od tih brojeva. Nakon toga se svi neprecrtani brojevi prepisuju na novi papir u istom poretku, a Ana dobija bodove tako što sabira proizvode dva susjedna broja. Na primjer, ako je prepisani niz {3, 6, -1, 2 }, tada Ana dobija 3*6 + 6*(-1)+(-1)*2 = 10 bodova. Ukoliko prepisani niz ima manje od 2 broja, Ana dobija nula bodova. Ana je u nedoumici: ne zna koje brojeve treba precrtati na Dinovom spisku tako da ostvari maksimalni mogući broj bodova. Napišite program koji će odrediti taj maksimalni mogući broj bodova.

Ulazni podaci

U prvom redu se nalazi broj N ($3 \leq N \leq 30\,000$), koliko ima brojeva zapisanih na Dinovom papiru. U svakom od sljedećih N redova nalazi se po jedan realan broj Xi ($-10.00 \leq Xi \leq 10.00$) sa Dinovog papira, zapisan sa tačno dvije decimale, sa nulama na kraju ako je potrebno.

Izlazni podaci

U prvi i jedini red štampati maksimalni broj bodova koje Ana može dobiti. Rezultat je potrebno štampati sa tačno četiri decimalna mjesta.

Test primjeri

Ulaz	Izlaz
3 8.52 1.00 9.61	81.8772
5 1.00 2.00 3.00 4.00 5.00	40.0000
3 -1.25 10.00 -1.00	1.2500

Rješenje:
Ovaj zadatak se rješava dinamičkim programiranjem. Definišemo niz dp{i} kao najveću vrijednost niza kod kojeg posljednji član se nalazi na poziciji i. Sada je lako naći rekurzivnu formulu (A1, A2, …, AN} je vrijednost ulaznog niza):

$$dp_0 = 0$$
$$A_0 = 0$$
$$dp_i = \max_{j<i}\{A_i \cdot A_j + dp_j\}$$

Ova implementacija ima složenost O(N^2), što nije dovoljno brzo. Sada ćemo primijetiti da mogućih vrijednosti elemenata ima veoma malo: tačnije, ima ih samo 2001 (brojeva sa tačno dvije decimale iz intervala [-10.00, +10.00]). Prepravimo algoritam na sljedeći način: sekvencijalno ćemo izračunavati vrijednosti niza dp za sve članove od 1 do N; izračunavanje same vrijednosti ubrzavamo tako što umjesto da biramo mjesto j na koje će se taj niz nastavljati, biramo vrijednost posljednjeg mjesta A_j. Sada je jedini član u gornjoj rekurzivnoj formuli koji ne znamo dp_j , ali u formuli vidimo da nas jedino zanima mjesto na kojem je taj dp_j maksimalan. Zbog toga ćemo uvesti pomoćni niz B_x koji će nam govoriti maksimalnu do sada viđenu vrijednost dp_z takvu da je A_z = x. Sada rekurzivna formula postaje:

$$dp_i = \max_{x\in[-10.00,+10.00]}\{A_i * x + B_x\}$$

Gornja formula nam omogućava implementaciju složenosti O(N*2001).

```
#include <algorithm>
#include <functional>

#include <cstdio>
#include <cstdlib>
#include <cstring>

#include <string>
#include <vector>

using namespace std;

const int MAXN = 30100;
const int MAXVAL = 2001;

int n, A[ MAXN ];

long long dp[ MAXN ];
long long Maks_dp[ MAXVAL ];

char Ima[ MAXVAL ];
```

```
int main( void )
{
    scanf( "%d", &n );

    for( int i = 0; i < n; ++i ) {
        double x; scanf( "%lf", &x );
        int sign = 1; if( x < 0 ) { sign = -1; x = -x; }
        A[i] = sign * int( 100*x + 1e-9 );
    }

    dp[0] = 0;
    memset( Ima, 0, sizeof Ima );
    memset( Maks_dp, 0, sizeof Maks_dp );
    Ima[ A[0]+1000 ] = true;

    long long maks = 0;

    for( int i = 1; i < n; ++i ) {
        dp[i] = 0;

        for( int j = 0; j < MAXVAL; ++j ) {
            if( Ima[j] == false ) continue;
            long long val = A[i]*(j-1000) + Maks_dp[j];
            if( val > dp[i] ) dp[i] = val;
        }

        Ima[ A[i]+1000 ] = true;
        if( dp[i] > Maks_dp[ A[i]+1000 ] ) Maks_dp[ A[i]+1000 ] =
dp[i];
        if( dp[i] > maks ) maks = dp[i];
    }

    printf( "%.4lf\n", double( maks ) / 10000.0 );
    return (0-0);
}
```