Kantonalno takmičenje iz informatike za učenike srednjih škola KS 2017. godine

Sarajevo, 15. april 2017.

Papiri

U ovom zadatku vaš problem je vrlo jednostavan: dobili ste jedan svežanj papira pri čemu je na svakom napisan neki cijeli broj ispred kojeg se nalazi znak plus ili minus – dakle niz cijelih brojeva. Na prvom papiru nalazi se poruka:

"Trebate sračunati najveći zajednički djelilac brojeva u skupu u slučaju bilo kakve promjene skupa. Plus označava dodavanje broja u skup, a minus označava izbacivanje broja iz skupa. Ako je skup prazan najveći zajednički djelilac je 1."

Npr. na papirima ste vidjeli brojeve:

+30
+15
+3
+5
-5
-15

Nakon svakog papira rješenje bi glasilo: 30, 15, 3, 1, 3, 3. U nastavku ste uočili da ima još papira, pa ste zaključili kako bi bilo dobro napraviti program koji će ispisivati rezultat umjesto vas.

Format ulaza

U prvom redu se nalazi prirodan broj N koji označava broj papira. U svakom sljedećem redu nalazi se cijeli broj ispred kojeg se nalazi znak plus ili minus.

Format izlaza

Izlaz treba da sadrži tačno N brojeva koji predstavljaju NZD skupa nakon unosa N-tog papira.

Ograničenja

Pazite da se program ne izvršava duže od 5s. Broj papira može biti do 10^5.

Primjer

Ulaz:	*Izlaz:*
6 +30 +15 +3 +5 -5	30 15 3 1 3 3

-15	

Rješenje (C++):

Sljedeće naivno rješenje bi davalo tačno rješenje, ali ne bi se uspjelo završiti u predviđenom vremenu za oko 50% testova.

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

int main ()
{
   fstream fulaz,fizlaz;
   fulaz.open("ulaz.txt", fstream::in);
   fizlaz.open("izlaz.txt",fstream::out);
   int collection [100000];
   int q = 0;
   fulaz>>q;
   int k = 0;
   char sign;
   int x;
   for(int i=0;i<q;i++){
      fulaz>>sign;
      fulaz>>x;
      if(sign=='+'){
         collection[k]=x;
         k++;
      }
      else{
         int j=0;
         while(true){
            if(collection[j]==x){
               for(int t=j;t<k-1;t++){
                  collection[t]=collection[t+1];
               }
               k--;
               break;
            }
            j++;
         }
      }
      int max = 2;
      int divisor = 1;
      int chosenDivisor;
      if(k==0){ fizlaz<<1<<endl;}
      else{
         while(divisor<=max){
            bool divisable = true;
            for(int j=0;j<k;j++){
               if(collection[j]%divisor!=0){
                  divisable = false;
                  break;
```

```
            }
            if(max<collection[j]) max = collection[j];
          }
          if(divisable == true){
            chosenDivisor = divisor;
          }
          divisor++;
        }
        fizlaz<<chosenDivisor<<endl;
      }
    }
    return 0;
}
```

Ispod je rješenje koje ima zadovoljavajuće performanse:

```
#include <iostream>
#include <fstream>
using namespace std;
const unsigned int N=1<<18;
const int n=18;
int a[N];
int tren=0;
int t[N][n+1];
int adds=0,rems=0;
int f(int a,int b){//NZD
    int r,i;
    if(a==0||b==0) return (b>a)?b:a;
    while(b!=0){
        r=a%b;
        a=b;
        b=r;
    }
    return a;
}
void init() {
  for (unsigned int x = 0; x < N; x++){
    a[x]=0;
    t[x][0] = a[x];
    }

  for (int y = 1; y <= n; y++)
    for (int x = 0; x < N; x+=(1<<y))
      t[x][y] = f(t[x][y-1], t[x+(1<<(y-1))][y-1]);
}
void postavi(int x, int v) {
  t[x][0] = a[x] = v;
  for (int y = 1; y <= n; y++) {
    int xx = x-(x&((1<<y)-1));
    t[xx][y] = f(t[xx][y-1], t[xx+(1<<(y-1))][y-1]);
  }
}
```

```
void dodaj(int v){
   tren++;
   adds++;
   for(unsigned int x=0;x<N;x++){
      if(a[x]==0){postavi(x,v); return;}
   }
}
void izbaci(int v){
   rems++;
   for(unsigned int x=0;x<N;x++){
      if(a[x]==v){postavi(x,0); return;}
   }
}

int vrati() {
   if(adds==rems)return 1;
 unsigned int i=0,j=tren+1;
 int res = 0, h = 0;
 while (i+(1<<h) <= j) {
  while ((i&((1<<(h+1))-1)) == 0 && i+(1<<(h+1)) <= j) h++;
  res = f(res, t[i][h]);
  i += (1<<h);
 }
 while (i < j) {
  while (i+(1<<h) > j) h--;
  res = f(res, t[i][h]);
  i += (1<<h);
 }
 return res;
}
int main()
{
   init();
   fstream fulaz,fizlaz;
   fulaz.open("ulaz.txt", fstream::in);
   fizlaz.open("izlaz.txt",fstream::out);
   int z=0;
   char zn;
   int tv;
   fulaz>>z;
   for(unsigned int i=0;i<z;i++){

      fulaz>>zn>>tv;
      if(zn=='+')dodaj(tv);
      else izbaci(tv);
      fizlaz<<vrati()<<"\n";
   }
   return 0;
}
```

Testni primjeri nisu navedeni zbog manjka prostora.

Aritmetički izraz

Napišite program koji provjerava da li je uneseni tekst validan aritmetički izraz prema pravilima datim ispod te ispisuje DA ako jeste, a NE ako nije validan.

Aritmetički izraz se sastoji od:

- aritmetičkih operatora: + - * /
- brojeva: cijelih i realnih (kod realnih brojeva koristi se decimalna tačka, kao u engleskom jeziku, a brojevi poput "3." i ".2" se smatraju validnim, pa je čak i "." ispravan broj)
- izrazi mogu biti zatvoreni u male (okrugle) zagrade
- izraz u malim zagradama može se nalaziti isključivo unutar srednjih (uglastih) zagrada, a srednje zagrade unutar velikih (vitičastih) zagrada; više od tri nivoa ugnježdenih zagrada nije dozvoljeno;
- svaki operator mora imati dva operanda; izuzetak je unarni minus (za promjenu predznaka) koji nema lijevog operanda

Svi ostali karakteri u stringu su zabranjeni. Program treba provjeravati i: da li se s obje strane svakog operatora nalaze ili brojevi ili izrazi u zagradama, da li svaka otvorena zagrada ima svoju zatvorenu, da li se ukrštaju, te da izraz u zagradama mora sadržavati barem jedan operator koji nije u drugom paru zagrada (tako npr. string "[(2+3)]" nije validan izraz jer se unutar srednjih zagrada ne nalazi niti jedan operator koji nije u malim zagradama).

Format ulaza

U prvom redu se nalazi prirodan broj N koji označava broj izraza koje treba provjeriti. U narednih N redova nalaze se stringovi koji sadržavaju aritmetičke izraze za provjeru. Ovi stringovi neće nikada sadržavati razmake.

Format izlaza

Izlaz treba da sadrži tačno N linija pri čemu se u svakoj liniji nalazi ili DA ili NE.

Primjer

Ulaz:	*Izlaz:*
2 2+3 [(2+3)]	DA NE

Rješenje (C++):

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

using namespace std;

int provjeri_izraz(char* s) {
	int u_malim=0, male_op=0, u_srednjim=0, srednje_op=0, u_velikim=0, velike_op=0,
velike_srednje=0, srednje_male=0;
	int op_pocetak=0, op_kraj=0, iza_zagrade=0;

	while (*s != '\0') {
		if (*s == '(') {
			if (u_malim || op_pocetak) return 0;
			u_malim=1;
			op_kraj=0;
			if (u_srednjim) srednje_male=1;

		} else if (*s == ')') {
			if (!u_malim || !male_op || op_kraj) return 0;
			u_malim=male_op=0;
			op_pocetak=1;
			iza_zagrade=1;

		} else if (*s == '[') {
			if (u_srednjim || op_pocetak) return 0;
			u_srednjim=1;
			op_kraj=0;
			if (u_velikim) velike_srednje=1;

		} else if (*s == ']') {
			if (!u_srednjim || !srednje_op || !srednje_male || op_kraj || u_malim) return 0;
			u_srednjim=srednje_op=0;
			op_pocetak=1;
			iza_zagrade=1;

		} else if (*s == '{') {
			if (u_velikim || op_pocetak) return 0;
			u_velikim=1;
			op_kraj=0;

		} else if (*s == '}') {
			if (!u_velikim || !velike_op || !velike_srednje || op_kraj || u_malim || u_srednjim)
return 0;
			u_velikim=velike_op=0;
			op_pocetak=1;
			iza_zagrade=1;
```

```
        } else if (*s == '*' || *s == '+' || *s == '-' || *s == '/') {
            if (*s != '-' && (op_pocetak==0 || op_kraj==1)) {
                return 0;
            }
            if (u_malim) male_op=1;
            else if (u_srednjim) srednje_op=1;
            else if (u_velikim) velike_op=1;
            op_pocetak=0;
            op_kraj=1;
            iza_zagrade=0;

        } else if (*s >='0' && *s <= '9' || *s == '.') {
            if (iza_zagrade) return 0;
            if (op_pocetak==0) op_pocetak=1;
            if (op_kraj==1) op_kraj=0;

        } else {
            //printf ("Nepoznat karakter %c\n", *s);
            return 0;
        }
        s++;
    }
    if (u_malim || u_srednjim || u_velikim || op_kraj) return 0;
    return 1;
}

int main() {
    int broj_izraza, i;
    string izraz;

    cin >> broj_izraza;
    cin.ignore(1000, '\n');

    for (int i=0; i<broj_izraza; i++) {
        getline(cin, izraz);
        char *s = (char*)izraz.c_str();
        if (provjeri_izraz(s)) cout << "DA\n"; else cout <<"NE\n";
    }
    return 0;
}
```

Testni primjeri

Ulaz:	*Očekivani izlaz programa:*
Primjer 1: 4 1+2 123456 0 aaaa **Primjer 2:** 5	**Primjer 1:** DA DA DA NE **Primjer 2:** NE

5*4aaa	DA
.+.	NE
1.2+%3.4	NE
13^14	NE
4,5+6,7	
Primjer 3:	**Primjer 3:**
5	DA
184/562	NE
15++15	DA
6*-3	NE
+2-4	NE
15*20/	
Primjer 4:	**Primjer 4:**
3	DA
(3+2)	NE
()	NE
[(1+1)]	
Primjer 5:	**Primjer 5:**
5	DA
(3+2)-(15*16)	DA
4-(6*7)	DA
(1287.4+1)/2	NE
(14+14)(3/2)	NE
(2*(1+1))	
Primjer 6:	**Primjer 6:**
5	DA
1-[(4+3)*7]	DA
{(8/7)-[(4+3)*7]}+4	NE
{(8/7)-[(4+3)*7]}{}	NE
123(3+4)	NE
(18.4-3)4	
Primjer 7:	**Primjer 7:**
4	NE
[2+3]	NE
{18/23+[4*4]}	DA
[2+(1+2)]	NE
(182/[4*(4-2)]+1)	
Primjer 8:	**Primjer 8:**
5	NE
[180*(17-3])	NE
([150-2)/4]	DA
[(150-2)/4]	NE
{[154-(4+2]/4)*8}	DA
{[154-(4+2)/4]*9}	
Primjer 9:	**Primjer 9:**
7	NE
1*(2+2	NE
[(4-8)*2	NE
{143-[2*2+(4/2]}	DA
{143-[2*2+(4/2)]}	NE
{143-[2*2+(4/2)]	DA
[(4-8)*2]	NE
[(4-8*2]	
Primjer 10:	**Primjer 10:**
5	NE

3+2)	NE
(4-8)*2]	DA
[(4-8)*2]	NE
[4-8)*2	NE
143-[2*2+(4/2)]}	

Run-Lenght Encoding (rle)

Jedna od najčešćih aplikacija na računaru su programi za kompresovanje (ZIPovanje) datoteka. Najjednostavniji metod kompresije je RLE kompresija kod koje se uočava određeni niz znakova koji se ponavljaju i zamjenjuje se posebnom oznakom ponavljanja i brojem ponavljanja.

Napišite program u kojem korisnik unosi string a zatim se vrši RLE kompresija stringa po sljedećem principu: ukoliko se naiđe na neki znak (char) koji se ponavlja više od tri puta potrebno ga je zamijeniti sljedećim nizom znakova: uzvičnik (!) koji označava da slijedi ponavljanje, zatim znak koji se ponavlja i na kraju broj puta koliko se znak ponavlja. Ovaj broj treba biti predstavljen znakovima koji odgovaraju ciframa. Ukoliko se u primljenom stringu naiđe na znak uzvičnik, potrebno ga je zamijeniti sa dva uzvičnika kako ne bi bio pogrešno protumačen kao znak kompresije. Radi lakšeg lančanog pozivanja funkcija vraća pokazivač na prvo slovo primljenog stringa.

Primjer: Ako je dat string

aaaaabbb!ccccccccccc

potrebno ga je zamijeniti sa

!a5bbb!!!c12

Objašnjenje:

- !a5 – znak a se ponavlja pet puta
- bbb – pošto se znak b ponavlja samo tri puta, nećemo ga kompresovati jer se time ništa ne dobija
- !! - znak uzvičnik je ponovljen (escapovan) da ne bi bio pogrešno protumačen kao znak kompresije
- !c12 – znak c se ponavlja 12 puta.

Cifre se uvijek kompresuju, čak i ako se pojavljuju samo jednom, da bi se izbjegla zabuna sa brojem ponavljanja. Npr. ako string glasi:

aaaaa4

bez ovog pravila kompresovani string bi glasio

!a54

a to bi značilo da se slovo a ponavlja 54 puta, pa zato kompresujemo i cifru 4 tako da dobijamo:

!a5!41

Format ulaza

Na ulazu se nalazi tekst koji treba kompresovati, koji može sadržavati sve dozvoljene ASCII karaktere.

Format izlaza

Na izlazu se nalazi kompresovani tekst

Primjer

Ulaz:	*Izlaz:*
`aaaaa4`	`!a5!41`

Rješenje (C):

```c
#include <stdio.h>

int main() {
	char c, prethodni='\0';
	int broj, i;
	do {
		c = getchar();
		if (c == prethodni) {
			broj++;
		} else {
			if (prethodni == '!')
				for (i=0; i<broj; i++) printf("!!");
			else if (prethodni >= '0' && prethodni <= '9' || broj > 3)
				printf("!%c%d", prethodni, broj);
			else
				for (i=0; i<broj; i++) printf("%c", prethodni);
			prethodni=c;
			broj=1;
		}
	} while(!feof(stdin));
	return 0;
}
```

Testni primjeri

Ulaz:	*Očekivani izlaz programa:*
Primjer 1: `aaaaabbb!ccccccccccc`	**Primjer 1:** `!a5bbb!!!c12`

Primjer 2: !a5bbb!!!c12 **Primjer 3:** Ovo je neki primjer teksta. wwwxxxyyzzz **Primjer 4:** !@@@@##$$$$$$%%%%^^^^^ ^^^&&&&&& ****((()))))__++++0000 111112222 3345555666777777788889 **Primjer 5:** Tekstaaaaa a................ **Primjer 6:** !!!!!!!!!!!!!!!!!!!!!!	**Primjer 2:** !!a!51bbb!!!!!!c!11!21 **Primjer 3:** Ovo je neki primjer teksta. wwwxxxyyzzz **Primjer 4:** !!!@4##!$6!%4!^8!&6!*4(((!)5__!+4!04!15!24!3 2!41!54!63!77!84!91 **Primjer 5:** ! 10Tekst!a6!.16 **Primjer 6:** !!

Backup

Na disku računara može se nalaziti više fajlova (datoteka) sa istim imenom pod uslovom da se nalaze u različitim folderima (direktorijima). Da biste jednoznačno označili fajl na disku zajedno sa folderom u kojem se nalazi, koristi se putanja. Na Windows operativnom sistemu prvi disk ili particija označen je slovom C:, drugi slovom D: itd. Recimo da na disku D: imamo folder Dokumenti, pa u njemu folder Skola, pa u njemu Informatika i konačno u tom folderu imamo fajl zadatak2.cpp. Kompletna putanja do ovog fajla bila bi:

D:\Dokumenti\Skola\Informatika\zadatak2.cpp

Vidimo da su dijelovi putanje razdvojeni znakom backslash (obrnuta kosa crtica).

Sara redovno radi backup svog diska C: na eksterni disk koji, kada je prikopčan, biva označen slovom D:. No kopiranje kompletnog sadržaja diska C: na D: dugo traje. Pomozite Sari tako što ćete napraviti program koji uzima spisak svih fajlova na C: i na D:, a ispisuje one fajlove koji se nalaze na disku C: a ne nalaze se na D:

Pri rješavanju problema nije dozvoljeno koristiti sortiranje!

Format ulaza

U prvom redu ulaza nalazi se prirodan broj N koji je broj fajlova na disku D:. U narednih N redova nalazi se N putanja. Nakon toga unosi se broj M koji predstavlja broj fajlova na disku C: te M putanja.

Format izlaza

Na izlazu se trebaju nalaziti putanje fajlova na disku C: koji se ne nalaze na disku D:

Ograničenja

Na disku C: može se nalaziti najviše 10^6 fajlova a na disku D: najviše 10^5 fajlova. Program ne smije raditi duže od 10 sekundi te ne smije koristiti više od 1 MB RAM. Nije dozvoljeno koristiti sortiranje – niti bibliotečnu funkciju niti vlastitu implementaciju.

Primjer

Ulaz:	*Izlaz:*
2 D:\Program Files\CodeBlocks D:\Windows\System 3 C:\Windows\System C:\Program Files\CodeBlocks	C:\Windows\NOTEPAD.EXE

C:\Windows\NOTEPAD.EXE	

Rješenje (C++):

Pošto je u zadatku navedeno ograničenje količine memorije te zabrana sortiranja, naivno rješenje bi bilo da se najprije u memoriju učitaju sve datoteke na disku D:, a da se zatim svaka sljedeća datoteka na disku C: pretražuje u ovom nizu.

No ovo rješenje ne bi radilo dovoljno brzo za otprilike 50% testova. Naivno rješenje dato je ispod.

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

using namespace std;

int main() {
	vector<string> disk_d;
	string s;

	int broj_D;
	cin >> broj_D;
	cin.ignore(1000, '\n');

	for (int i(0); i<broj_D; i++) {
		getline(cin, s);
		if (s[0] == 'D') s[0] = 'C';
		disk_d.push_back(s);
	}

	int broj_C;
	cin >> broj_C;
	cin.ignore(1000, '\n');

	for (int i(0); i<broj_C; i++) {
		getline(cin, s);
		bool nadjen(false);
		for (int j(0); j<broj_D; j++) {
			if (disk_d[j] == s) {
				nadjen=true;
				break;
			}
		}
		if (!nadjen)
			cout << s << endl;
	}

	return 0;
}
```

Rješenje koje prolazi 100% testova moralo bi koristiti neku metodu optimizacije. Jedna od tih metoda je da prepoznamo da folderi na disku čine stablo te da koristimo podatkovnu strukturu n-arnog stabla, a ovakvo rješenje je dato ispod. Stablo je implementirano kao vektor čvorova. Rješenje je nešto duže radi lakšeg razumjevanja.

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

using namespace std;

struct Cvor {
	string naziv;
	int sljedeci;
	int dijete;
};

vector<Cvor> stablo;
int korijen, velicina, kapacitet;

void stavi_u_stablo(string s, int poc, int trenutni, int roditelj) {
	// Trazimo backslash
	int kraj(s.find('\\', poc));
	if (kraj==-1) kraj=s.length();
	string naziv_cvora = s.substr(poc, kraj-poc);

	// Ima li Cvor sa tim nazivom?
	int prethodni(-1);
	while (trenutni != -1) {
		if (stablo[trenutni].naziv >= naziv_cvora) break;
		prethodni = trenutni;
		trenutni = stablo[trenutni].sljedeci;
	}

	// Ima!
	if (trenutni > -1 && stablo[trenutni].naziv == naziv_cvora) {
		// Nastavljamo sa sljedećim blokom u stringu
		if (kraj<s.length())
			stavi_u_stablo(s, kraj+1, stablo[trenutni].dijete, trenutni);
		return;
	}

	// Nema - potrebno dodati novi
	Cvor c;
	c.naziv = naziv_cvora;
	c.sljedeci = -1;
	c.dijete = -1;
	stablo[velicina++] = c;
	if (velicina == kapacitet) {
		kapacitet += 1000;
		stablo.resize(kapacitet);
```

```
	}

	// Prvi cvor u stablu
	if (velicina == 1) {
		korijen = 0;
	}

	// Roditelj nema djece
	else if (trenutni == -1 && prethodni == -1) {
		stablo[roditelj].dijete = velicina-1;
	}

	// Novi korijen
	else if (prethodni == -1 && roditelj == -1) {
		stablo[velicina-1].sljedeci = korijen;
		korijen = velicina-1;
	}

	// Prvi u listi
	else if (prethodni == -1) {
		stablo[velicina-1].sljedeci = stablo[roditelj].dijete;
		stablo[roditelj].dijete = velicina-1;
	}

	// Ubacujemo u sredinu ili na kraj liste
	else {
		stablo[velicina-1].sljedeci = trenutni;
		stablo[prethodni].sljedeci = velicina-1;
	}

	// Ako ima još dijelova, dodajemo ih ispod upravo dodatog
	if (kraj<s.length())
		stavi_u_stablo(s, kraj+1, -1, velicina-1);
}

void pisi_cvor(int trenutni, int dubina) {
	if (trenutni==-1) return;

	for (int i(0); i<dubina; i++) cout << "   ";
	cout << stablo[trenutni].naziv << endl;

	pisi_cvor(stablo[trenutni].dijete, dubina+1);
	pisi_cvor(stablo[trenutni].sljedeci, dubina);
}

bool trazi(string s, int poc, int trenutni) {
	if (trenutni == -1) return false;

	int kraj(s.find('\\', poc));
	if (kraj==-1) kraj=s.length();
	string naziv_cvora = s.substr(poc, kraj-poc);
```

```
        // Ima li Cvor sa tim nazivom?
        while (trenutni != -1) {
            if (stablo[trenutni].naziv == naziv_cvora) {
                if (kraj == s.length()) return true;
                return trazi(s, kraj+1, stablo[trenutni].dijete);
            }
            if (stablo[trenutni].naziv > naziv_cvora) return false;
            trenutni = stablo[trenutni].sljedeci;
        }
        return false;
}

int main() {
        string s;

        // Pocetni parametri stabla
        korijen=-1; velicina=0; kapacitet=1000;
        stablo.resize(kapacitet);

        int broj_stringova;
        cin >> broj_stringova;
        cin.ignore(1000, '\n');
        bool svi_nadjeni = true;

        for (int i(0); i<broj_stringova; i++) {
            getline(cin, s);
            if (s[0] == 'D') s[0] = 'C';
            stavi_u_stablo(s, 0, korijen, -1);
        }

        cin >> broj_stringova;
        cin.ignore(1000, '\n');

        for (int i(0); i<broj_stringova; i++) {
            getline(cin, s);
            if (!trazi(s, 0, korijen)) {
                cout << s << endl;
                svi_nadjeni=false;
            }
        }
        //pisi_cvor(korijen, 0);

        return 0;
}
```

Testni primjeri

Zbog nedostatka prostora testni primjeri se ne mogu u cjelosti navesti, zbog toga ćemo ih ukratko opisati ispod:

Testni primjer 1: D: 10^4 fajlova, C: 10^4 fajlova, svi jednaki

Testni primjer 2: D: 10^4 fajlova, C: 10^4 fajlova, svi različiti

Testni primjer 3: D: 10^4 fajlova, C: 10^5 fajlova, prvih 10^4 jednaki

Testni primjer 4: D: 10^5 fajlova, C: 10^4 fajlova, prvih 10^4 jednaki

Testni primjer 5: D: 10^5 fajlova, C: 10^5 fajlova, svi jednaki

Testni primjer 6: D: 10^5 fajlova, C: 10^5 fajlova, svi različiti

Testni primjer 7: D: 10^5 fajlova, C: 10^6 fajlova, prvih 10^5 jednaki

Testni primjer 8: D: 10^5 fajlova, C: 10^6 fajlova, svi različiti