Pripreme pred okružno takmičenje 2021.

Zagrevanje

1.Dve niske su anagrami ako se sastoje od istog broja istih karaktera. Napisati program koji proverava da li su dve niske karaktera anagrami. Niske se zadaju sa standardnog ulaza i neće biti duže od 127 karaktera. Uputstvo: Napisati funkciju koja sortira slova unutar niske karaktera, a zatim za sortirane niske proveriti da li su identične.

ULAZ	IZLAZ
vatra trava	Jesu anagrami
vatra tavav	Nisu anagrami

Rešenje 01

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

int main()
{
    string s1, s2;
   getline(cin,s1);
   getline(cin,s2);

   sort (s1.begin(), s1.end());
   sort (s2.begin(), s2.end());

   if (s1==s2)
                    cout <<"Jeste anagram\n";
           else
                    cout <<"Nije anagram\n";


   return 0;
  }
```

Resenje 02

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

int main()
{
    string s1, s2;
   getline(cin,s1);
   getline(cin,s2);
   int n1, n2;
```

```
  n1=s1.length();
  n2=s2.length();
  sort (s1.begin(), s1.begin()+n1);
  sort (s2.begin(), s2.begin()+n2);

  if (s1==s2)
                cout <<"Jeste anagram\n";
        else
                cout <<"Nije anagram\n";


  return 0;
 }
```

Rešenje 03 (NEISPRAVNO)

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

int main()
{
   string s1, s2;
   getline(cin,s1);
   getline(cin,s2);
   int n1, n2;
   n1=s1.length();
   n2=s2.length();
   sort (s1, s1.begin()+n1);
   sort (s2, s2.begin()+n2);

   if (s1==s2)
      cout <<"Jeste anagram\n";
   else
      cout <<"Nije anagram\n";


   return 0;
}
```

2. Objasnite zašto sledeći program nije korektan.

```
#include <cstdio>
using namespace std;
#define Nmax 100
int a[Nmax], i;
int main()
{
   for (i = 0; i <= 100; i++) { a[i] = 0; }
   printf("OK\n");
   return 0;
}
```

Analizirajte adresni prostor promenljivih!!!

```
#include <cstdio>
using namespace std;
#define Nmax 100
int a[Nmax], i;
int main()
{
printf("Adresa promenljive i je %p\n", &i);
printf("Adrese a[0], a[1], a[99], a[100] su redom  %p %p %p %p \n", &a[0], &a[1], &a[99],
&a[100]);
   for (i = 0; i <= 100; i++) { a[i] = 0; }
   printf("OK\n");
   return 0;
}
```

Palindromi

3. Data je niska sastavljena od N (0 < N < 500 000) malih slova abecede. Niska nije palindrom. Dopisujemo sleva nisci nekoliko malih (1, 2, …) slova abecede tako da se dobije novi niz, koji je palindrom. Napišite programa koji ispisuje kolika je najmanja moguca dužina nove niske.

Primer
Ulaz
babc
Izlaz
5

Rešenje

Neka je data ulazna niska s="abbac"

Dovoljno je da postavimo poslednje slovo c napred i dobicemo nisku "cabbac", koje jeste palindrom s minimalna dužinom, i zadovoljava uslove zadatka.

Slican postupak važi i za opsti slučaj, koda ulazna niska s sadrži podniz na pocetku niske koji je palindrom.

Predstavimo ulaznu nisku u formi s = s1 + s2, gde s1 je najduži prefix od s, koji je palindrom.

Tada možemo da sastavimo niz t = s3 + s1 + s2, gde s3 je inverz od s2.

Kako niska s1 je maksimalno dugacak palindrom, onda niska t je palindrom, napravljen po uslovima zadatka i to najkraći moguć.

Dužina niske t je:

L = n2+n1+n2=2*n2 + n1,

gde n1 je duzina za s1, n2 je dužina za s2 (naravno i za s3). Polazna dužina niske s je n tj. n = n1 + n2. Dakle, L = 2*(n-n1) + n1 = 2*n – n1 i program mora da ispise L.

Da nadjemo n1, treba redom da se provere svi prefixi od niske s i da otkrijemo izmedju njh najduzi palindrom. Njegova dužina je n1.

Algoritam složenosti $O(n^2)$

```
#include<iostream>
using namespace std;
string s;
```

```
bool najveciPrefixPalindrom(int n,string &s) //provera da li je palndrom p duzine n
{
  for(int i=0;i<n/2;i++)
    if(s[i]!=s[n-i-1]) return false;
  return true;
}

int main()
{
  cin >> s;
  int n=s.size();
  for(int i=n-1;i>=0;i--)
     if(najveciPrefixPalindrom(i,s))  //trazimo najduzi prefix niske s koji moze biti palindrom
     {
       cout << 2*n-i << endl;
       return 0;
     }
}
```

Algoritam složenosti O(n)

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

string s;
const int Nmax=500002;
int KMPtabela[2*Nmax+2];

void KMP(int n)
{
  int j = 0;
  for (int i = 1; i < n; i++)
  {
    while (j > 0 && s[j] != s[i]) j = KMPtabela[j-1];
    if (s[j] == s[i]) j++;
    KMPtabela[i] = j;
  }
}

int main()
{
  cin >> s;
  int n=s.size();
  string sr = s;
  reverse(sr.begin(),sr.end());
  s=s+"#"+sr;
  int nn=s.size();
  KMP(nn);
  int n1=KMPtabela[nn-1];
  cout << 2*n-n1 << endl;
```

```
  cout << endl;
}
```

4. Reči

Mali Perica je nedavno krenuo u školu i već je naučio da čita. Pošto je Perica veoma vredan i savestan učenik, od svoje učiteljice Marije zatražio je neki zadatak gde bi mogao da proveri svoje znanje. Učiteljica Marija, srećna što ima tako dobrog učenika, odmah mu je dala jedan problem na kojem će Perica proveriti koliko dobro je savladao gradivo.
Perica je dobio mali rečnik R, koji sadrži n reči. Pored toga, učiteljica mu je dala i jednu listu L od m reči. Zadatak mu je da testira svaku reč iz liste L na sledeći način: Za svaku reč iz rečnika pronaći će sva njena pojavljivanja u reči koju testira i precrtaće svako pojavljivanje. Na kraju, Perica mora da saopšti svojoj učiteljici koliko reči iz liste ima svako svoje slovo precrtano bar jedanput.
Pomozite učiteljici Mariji tako što ćete napraviti program koji određuje koliko reči iz liste L će biti potpuno precrtano, kako bi ona lakše proverila da li je Perica dobro uradio domaći zadatak.
Ulaz:
U prvom redu nalazi broj n ($1 \leq n \leq 100$), broj reči u rečniku R. U sledećem redu nalazi se n reči iz rečnika. Reči su razdvojene jednim razmakom i nisu duže od 10 slova. U trećem redu nalazi se broj m ($1 \leq m \leq 100$), broj reči u listi L.
Narednih m redova sadrže po jednu reč iz liste L. Svaka reč iz liste neće biti duža od 100 slova. Sva slova su mala slova engleskog alfabeta.
Izlaz:
U prvom i jedinom redu izlaza potrebno je ispisati broj potpuno precrtanih reči.
Primer:
Ulaz
6
paja ram papa ma lama lig
7
pa~~paja~~
salama
papaje
~~ram~~alama
mara
mama
milig~~ram~~

Izlaz
3

Objašnjenje.
Ima 3 potpuno precrtane reči: papaja, ramalama, mama.
Memorijsko ograničenje 64MB.
Vremensko ograničenje 1s.

Rešenje
Problem Reči može da se reši direktnom primenom koraka koji su opisani u formulaciji problema. Dakle, svaka rec iz liste L se posmatra zasebno. Neka je tekuca rec iz liste L oznacena sa word.

Algoritam obilaska se primenjuje nad svakom od reči iz ove liste.

Sa |w| označićemo dužinu reči w, kako iz rečnika R tako i iz liste L. Definisaćemo pomoćni niz (vector) ***precrtaj*** koji ce biti tipa bool.

Ovaj niz ce evidentirati sva gore opisana precrtavanja. Za tip uzimamo logicku vrednost, jer nam je samo bitno da li je slovo precrtano ili ne, a nije bitno koliko puta je precrtano. Na pocetku ni jedno slovo nije precrtano, odnosno niz ***precrtaj*** je inicijalizovan na false. Sada redom obilazimo reqi iz reqnika. Svaku req t ∈ R pokuxavamo da ”postavimo” u word na svim pozicijama. Drugim reqima, za svaku poziciji i, i ∈ [1,|word|−|t|+ 1], ispitujemo da li se podreq wordi wordi+1... wordi+|t|−1 poklapa sa t. Ukoliko se poklapa, u nizu mark elemente na pozicijama {i,i + 1,...,i + |t|− 1} postavljamo na true (jer smo ih precrtali pomo#u reqi t iz reqnika).
Nakon obilaska svih reqi iz reqnika, potrebno je ispitati da li su sva slova precrtana bar jednom. Ovo je ekvivalentno ispitivanju da li su svi elementi niza mark postavljeni na true. Ukoliko jesu, req word je potpuno precrtana, inaqe nije.

KMP optimizacija

Problem se mo#e uraditi i u slo#enosti O(m·n·100) ukoliko za upore#ivanje koristimo algoritam KMP (eng. Knuth Morris Pratt). Me#utim, ovde treba biti pa#ljiv pri markiranju jer ukoliko bi se ovo radilo kao u opisanom algoritmu slo#enost bi ostala ista. Primera radi ukoliko je word = aaa...aa i t = aa...aa tada svaku poziciju u nizu mark postavljamo na true |t| puta umesto po jednom. Kada ispitujemo konkretnu req t, pamtimo samo segmente koje treba markirati u nizu mark. Tek nakon ispitivanja svih pozicija vrxi se markiranje ali tako da se svaki element niza mark postavi najvixe jednom (sortiramo po levim krajevima i dok se kre#emo po nizu pamtimo najdalji desni kraj).

100% resenje bez KMP-a O(m·n·100·10)

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

using namespace std;



struct rec{
        string v;
        vector<bool> precrtaj;

        rec(string s)
        {
                v=s;
                precrtaj.resize(s.size(),0);
        }

        rec()
        {
                v="";
        }

        void test()
        {
                for(int i=0;i<v.size();i++)
                        cout<<precrtaj[i];
                cout<<endl;
        }


};
```

```
int n;
vector<string> r;
int m;
vector<rec> l;

void precrtaj(string &a,rec &b)
{

        int t=-1;
        bool moze=true;
        while(moze)
        {
                t=b.v.find(a,t+1);
                if(t!=string::npos)
                {
                        for(int i=0;i<a.size();i++)
                        {
                                b.precrtaj[t+i]=true;
                        }
                }
                else
                moze=false;
        }

}

int main()
{
        cin>>n;
        r.resize(n);
        for(int i=0;i<n;i++)
         cin>>r[i];

        cin>>m;
        for(int i=0;i<m;i++)
         {
                string s;
                cin>>s;

                l.emplace_back(s);
         }

        for(int i=0;i<n;i++)
         for(int j=0;j<m;j++)
         {
                precrtaj(r[i],l[j]);
         }

         int br=0;
        for(int i=0;i<m;i++)
        {
                bool moze=true;
                for(int j=0;j<l[i].precrtaj.size();j++)
                        moze = moze && l[i].precrtaj[j];

                if(moze)
                  br++;
        }
        cout<<br<<endl;




  return 0;
}
```

100% resenje

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
int n;
char r[200][200];
int p[200];
int m;
char l[200][200];
bool t[200][200];
int k[200];
int main()
{
	scanf("%d",&n);
	for(int i=0;i<n;i++)
	{
		scanf("%s",r[i]);
		p[i]=strlen(r[i]);
	}
	scanf("%d",&m);
	for(int i=0;i<m;i++)
	{
		scanf("%s",l[i]);
		k[i]=strlen(l[i]);
	}
	bool ok;
	bool rec;
	int v=0;
	int g=0;
	for(int i=0;i<m;i++)
	{
		rec=true;
		g=0;
		for(int j=0;j<k[i];j++)
		{
			for(int s=0;s<n;s++)
			{
				ok=true;
				if(p[s]+j<=k[i])
				{
					for(int o=0;o<p[s];o++)
					{
						if(l[i][j+o]!=r[s][o])
						{
							ok=false;
							o=p[s];
						}
					}
					if(ok)
					{
						for(int o=0;o<p[s];o++)
						{
							if(!t[i][j+o]) g++;
							t[i][j+o]=true;
						}
					}
				}
			}
			if(!t[i][j])
			{
				rec=false;
				j=k[i];
			}
			if(g==k[i])
			{
				j=k[i];
				rec=true;
			}
```

```
                }
                if(rec && g==k[i]) v++;
        }
        printf("%d",v);
        return 0;
}
```

MERENJE VREMENA

http://poincare.matf.bg.ac.rs/~jelenagr/1d/PretrageSortiranje.pdf

http://poincare.matf.bg.ac.rs/~jelenagr/P2/cas04/

Eksperiment sa razlicitim pretragama

Dimenzija niza: 1000000

```
        Linearna:   7786304 ns
        Binarna:               1019 ns
        Interpolaciona:         428 ns

Dimenzija niza: 100000
        Linearna:    698932 ns
        Binarna:                530 ns
        Interpolaciona:         192 ns

Dimenzija niza: 1000000
        Linearna:   6416484 ns
        Binarna:                616 ns
        Interpolaciona:         474 ns

Dimenzija niza: 100
        Linearna:       977 ns
        Binarna:                384 ns
        Interpolaciona:         340 ns
```

BATCH FILE za proveru vremena

provera.bat

```
echo  %TIME%
@echo  off
vikendica < vikendica20.in  > vikendica20.izl
echo.
echo  %TIME%
fc vikendica20.izl vikendica20.out
```

Memory layout i ispis zauzeca memorije

http://www.geeksforgeeks.org/memory-layout-of-c-program/

Česte pogreške

Donosimo listu čestih i nepotrebnih greški.

Sve od sledećeg može rezultirati time da će evaluator dodijeliti nula bodova vašem rešenju.

• Manjak inicijalizacije lokalnih varijabli:
U jezicima C i C++ lokalne varijable (što uključuje i varijable deklarisane unutar funkcije main) ne inicijaliziraju se automatski te njihova početna vrednost nije definisana. Obavezno inicijalizirajte vrednosti svih lokalnih varijabli, inače je moguće da vaš program dobro radi lokalno, a dobije nula bodova na graderu.

• Rešenje alocira premale nizove: Ako ne alocirate dovoljno velike nizove, moguće je da vaš program dobro radi za primere iz teksta zadatka, a dobije nula bodova na graderu.
Na primer, ako s ulaza trebate pročitati niz od najviše 1000 znakova, u jeziku C potrebno je alocirati polje od najmanje 1001 elementa (char s[1001]).

• Rešenje alocira prevelike nizove: Nepotrebno glomazni nizovi mogu uzrokavati da vaše rešenje prekorači memorijsko ograničenje i dobije nula bodova na evaluatoru.
Na primer, u jeziku C niz deklarisan sa int x[1024][1024][100] koristi 400MiB memorije.

Korištenje naredbi itoa (koje nema pod Linuxom) ili atoi (koja radi malo drugačije pod Linuxom). Umjesto njih preporučamo korištenje funkcija sscanf i sprintf.

• Rešenja čekaju na pritisnut taster nakon ispisa rezultata, npr. zbog naredbe system("pause").

• Rešenja ispisuju rezultate u pogrešnom formatu, npr. ispisuju dva broja svaki u svoj red umesto u istom redu.

• Rešenja ispisuju višak podataka na standardni izlaz, npr. debug informacije ili poruke poput “Upišite broj:”, “Rešenje je:” i slično.

• Glavna funkcija u jezicima C i C++ definisana je kao void main() ili nema naredbe return 0;

• Manjak potrebnih include direktiva u jezicima C i C++. Dodatno, ako vec koristite C i C++ savjetujemo da prilikom testiranja iz komandne linije obavezno uključite optimizacijsku opciju koja se koristi na graderu (-O2) te opcije koji upozoravaju na moguće probleme u kodu (primer -Wall -Wextra).

Niže je ilustracija takvog prevođenja jednog rešenja. Prvo upozorenje koje je prijavio kompajler je lažna uzbuna, dok drugo ukazuje na stvarnu grešku (neinicijalizirana varijabla) zbog koje je takmicar nepotrebno izgubio bodove

```
$ g++ -O2 -o poli.exe poli.cpp -Wall -Wextra
poli.cpp: In function ‘int main()’:
poli.cpp:16:20: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
  for (int i = 0; i < in.length(); ++i)
                  ^
poli.cpp:36:29: warning: ‘curr’ may be used uninitialized in this function [-Wmaybe-uninitialized]
  num[(curr == 0 ? 1 : curr) - 1] = val;
```