ISPITNI CENTAR

DRŽAVNO TAKMIČENJE 2025.

AUTOR/AUTORKA TESTA _

RECENZENT/RECENZENTKINJA _

PODGORICA, _ _ _ _ _ _ _ _ _ _ _ _ 20_ _ _ GODINE

SREDNJA ŠKOLA

PROGRAMIRANJE

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. Takmičari u C-u i C++-u moraju koristiti programski alat CodeBlocks. 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 „Vaša šifra“ moraju se nalaziti datoteke sa snimljenim izvornim kôdovima rješenja. Vaša rješenja možete upload-ovati na sistem za takmičenje, ako je dostupan, i provjeriti koliko bodova ste dobili. Nakon takmičenja, komisija će testirati vaša rješenja još jednom. Na kraju svakog zadatka dati su primjeri test podataka. Ti primjeri služe da bi vam tekst zadataka bio što je moguće jasniji i 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. 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.

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!

VAŽNO za jezik Java!

Ne kreirajte pakete za vaše zadatke, već koristite podrazumijevani (default) paket.

Zadaci

Zadatak	Zadatak1	Zadatak2	Zadatak3
Izvorni kôd	zadatak1.java zadatak1.pas zadatak1.c zadatak1.cpp	zadatak2.java zadatak2.pas zadatak2.c zadatak2.cpp	zadatak2.java zadatak2.pas zadatak2.c zadatak2.cpp
Memorijsko ograničenje	256MB	256MB	256MB
Vremensko ograničenje (po test podatku)	1 sekund	1 sekund	1 sekund
Ukupno bodova	**30**	**33**	**37**

Zadatak 1

Prirodan broj nazivamo zgodnim ako su mu svake dvije susjedne cifre različite parnosti. Za dati prirodan broj N, odrediti koji je njemu najbliži zgodni broj?

Napomena: Jednocifreni brojevi su zgodni. Udaljenost dva broja predstavlja apsolutnu vrijednost njihove razlike.

Ulaz:

Jedini red standardnog ulaza sadrži prirodan broj N koji ima najviše 1000000 cifara.

Izlaz:

U jedini red izlaza ispisati traženi najbliži zgodni broj. Ako postoje dva najbliža broja, ispišite prvo manji, pa veći broj i odvojite ih jednim razmakom.

Primjeri:

Ulaz	**Izlaz**
13	12 14
5801001	5810101

U test podacima vrijednim 50% bodova važi da je broj cifara manji od 9.

Rješenje:

Pronađimo u broju N prvu cifru s lijeva koja narušava alternirajući parnost, tj. iste je
parnosti kao i prethodna cifra. Imamo dvije mogućnosti: a) Promijenićemo ovu cifru, a sve cifre prije nje ostaće nepromijenjene. b) Promijenićemo neku cifru prije ove. Jasno je da se slučaj b) nikada neće isplatiti, jer želimo da cifre velike težine ostanu nepromijenjene.

Mogućnost A grana se na dvije pod mogućnosti: povećavanje i smanjivanje cifre. Ako je ova cifra 0, možemo je samo povećati; ako je 9, možemo je samo smanjiti; inače je možemo i povećati i smanjiti. Jasno je da se ne isplati povećati ili smanjiti cifru za više od jedan.

Nakon povećanja ili smanjenja cifre, šta učiniti sa narednim ciframa? Ako smo povećali cifru, dobijeni broj je sigurno veći od N pa, da bismo mu se približili, naredne cifre treba da budu što manje: 0101… ili 1010..., zavisno od parnosti povećane cifre. Analogno, ako smo smanjili cifru, dobijeni broj je sigurno manji od N pa, da bismo mu se približili, naredne cifre treba da budu što veće: 9898… ili 8989..., zavisno od parnosti smanjene cifre.

Naravno, ako smo cifru mogli i povećati i smanjiti, treba da od dva dobijena broja pronađemo onaj koji je bliži broju N.

Zadatak 2

Hari Poter je od profesora Dambldora dobio na čuvanje drevnu Knjigu čarolija. Knjiga sadrži čarolije sposobne da uklone određene djelove teksta, kao da nikada nisu ni postojali. U međuvremenu je Voldemort smislio novu magičnu riječ, koja je toliko moćna da bi uz nju bio nezaustavljiv. Srećom, magična riječ koju je Voldemort smislio je toliko dugačka da nije mogao da je zapamti već je napisao na papirić. Hari je uz pomoć svojih prijatelja Rona i Hermione uspio da dođe do papirića na kojem je Voldemor napisao svoju magičnu riječ.

Kada Hari sazna za Voldemorotovu zavjeru, nema izbora nego da iskoristi Knjigu čarolija. On čita Voldemorov papirić i počinje da ga proučava slovo po slovo. Kada god pročita neki dio teksta koji se poklapa sa čarolijom iz Knjige, odgovarajući dio teksta nestaje pod dejstvom magije. Hari nastavlja ovaj postupak do samog kraja niza, sve vrijeme pazeći na potencijalne zamke koje je Voldemort postavio. Ako se u bilo kom trenutku otkrije više magičnih riječi istovremeno, Hari će iskoristi čaroliju koja je prva zapisana u Knjizi.

Ulaz:
U prvom redu nalazi se niz karaktera, koji predstavlja Voldemorovu novu magičnu riječ. Niz može imati najviše 100000 karaktera.
U sledećem redu se nalazi N, broj čarolija u knjizi ($1 \leq N \leq 100$).
U narednih N redova nalaze se čarolije poređane redom kojim se pojavljuju u knjizi.
Svaka čarolija iz knjige ima dužinu manju od 10000 karaktera.
Svi karakteri predstavljaju velika slova Engleske abecede.

Izlaz:
Jedan red u kome je tekst koji je ostao na Voldemorovom papiriću nakon što je Hari završio sa magijom.

Primeri:

Ulaz	Izlaz
KUKUKAKIKUKAKEKKUKAKAKKUKAKUKAKU 2 KEK UK	KAKIKAKAKAKKAKAKU
HEATHLEDGER 2 HEATH LEDGER	
KAPKAPPAPAK 1 KAPPA	K

U test podacima vrijednim 40% poena važi da je dužina teksta koji čita Hari manja od 1000 karaktera, i da je zbir dužina svih čarolija u knjizi manji od 1000 karaktera.

Rješenje:

Ideja rješenja je da se efikasno implementira opisani algoritam. Program prolazi kroz početni string S, provjerava na svakoj poziciji da li se tu završava neka čarolija i, ako se završava, briše odgovarajući podstring.
Da bi se brisanje podstringa obavljalo efikasno, potrebno je koristiti strukturu podataka nalik steku, nazvaćemo je S', gde se čuva trenutno obrađivani dio stringa S. Kada program dođe do sledećeg karaktera iz S, taj karakter se dodaje (push) u S'. Kada se utvrdi da se na poziciji i završava neka čarolija, karakteri koji je čine izbacuju se iz steka jedan po jedan.

Za maksimalan broj poena, neophodno je vršiti efikasno poređenje stringova. Jedan od mogući načina je pomoću heširanja. Za naše rješenje koristili smo polinimijalnu heš funkciju. Heš vrijednost stringa *a* čiji su karakteri $a_0a_1a_2a_3...a_k$ računamo kao

$$H(a)=(a_0*p^0+a_2*p^1+a_3*p^3+\cdots+a_kp^k) mod\ M$$

gde su *p* i *M* prosti brojevi, pri čemu je $p\approx100$, $M\approx10^9$.

Ako u nizu *pref* čuvamo izračunate heš vrijednosti svih prefiksa teksta koji Hari čita do pozicije na kojoj se trenutno nalazi, možemo efikasno da vršimo poređenje sufiksa pročitanog teksta sa čarolijama koje je Dambldor dao Hariju u složenosti O(1). Sve što je potrebno je da uporedimo odgovarajuće heš vrijednosti. Heš vrijednost podstringa između pozicije *i* i *j* može da se izračuna efikasno koristeći se formulom

$$H(a_ia_{i+1}a_{i+2}...a_j)=(pref[j]-pref[i-1]\cdot p^{(j-i+1)})\ mod\ M$$

Ako ključ *k* odgovara podstringu $a_ia_{i+1}a_{i+2}...a_j$, tada će morati da važi da je $H(a_ia_{i+1}a_{i+2}...a_j) = p^i*H(k).$

Zadatak 3

Voldemor je zaštitio svoje horkukse izuzetno moćnim čarolijama. Kako bi Hari spasio svijet, mora uništiti svaki horkuks. Međutim, prije nego što to uradi, mora do njih doći tako što će proći sve Voldemorove čarolije.

Jedna od čarolija koju je Voldemor postavio funkcioniše na sljedeći način: Voldemor je stvorio korijen stabla. Korijen stabla ima indeks 1 i u njemu je upisana vrijednost 0. Sada će Voldemor postavljati različite upite, a Hari treba da brzo i tačno odgovori na sve upite kako bi stigao do horkuksa. Upiti mogu biti jednog od sljedećeg oblika:

- 1 X Y – Dodaje novi čvor u stablo. Roditeljski čvor novog čvora je čvor sa indeksom X (čvor sa indeksom X se sigurno već nalazi u stablu). Y ($1 \leq Y \leq 10^9$) predstavlja vrijednost koja je upisana u novom čvoru. Indeks novog čvora će biti najmanji pozitivan cio broj takav da se u stablu ne nalazi čvor sa tim indeksom. Na primjer, prvi upit ovog tipa će dodati čvor sa indeksom 2, sljedeći čvor sa indeksom 3, zatim sa indeksom 4 ....

- 2 X – Posmatramo sva **podstabla** čiji je korijen jedan od **sinova** čvora sa indeksom X. Za svako od tih podstabala izračunavamo **zbir vrijednosti svih čvorova u podstablu**. Vaš zadatak je da ispišete **najveći** takav zbir (tj. maksimalnu sumu vrijednosti čvorova) među svim podstablima koja imaju korijen u nekom od sinova čvora X.

Pošto Hari nije previše dobar sa matematičkim zagonetkama, a Hermiona nije tu da mu pomogne, obratio se vama za pomoć.

Ulaz:

Prvi red standardnog ulaza sadrži prirodan broj Q ($Q \leq 200000$) koji predstavlja broj upita koje će Voldemor postaviti. Narednih Q redova sadrže opise upita, po jedan u redu.

Izlaz:

Potrebno je da štampate odgovore za svaki od upita tipa 2.

Ulaz	**Izlaz**
7 1 1 3 2 1 2 2 1 2 5 2 1 1 1 4 2 1	3 0 8 8

Podzadaci: U test primjerima vrijednim 70% nijedan čvor u stablu nema više od 400 sinova.

Rješenje

Prvo treba da primijetimo da možemo da kreiramo čitavo stablo i postavimo vrijednosti svih čvorova na nula. Ovo možemo da uradimo jer nula ne utiče na rezultat upita drugog tipa.

Nakon ovoga problem izvršavanja upita tipa 1 smo transformisali na postavljanje odgovarajuće vrijednosti čvoru u stablu.

Pokrenemo dfs algoritam od korijena stabla (čvor sa indeksom 1), i zapišemo u niz vrijednosti koje su upisane u čvorovima prilikom ulaska u čvor, i prilikom izlaska iz njega. Možemo da primijetimo da je svako podstablo predstavljeno segmentom u nizu. Početak i kraj segmenta možemo da dobijemo na osnovu vremena prvog ulaska u čvor i vremena izlaska iz čvora. Sada je problem nalaženja sume podastabla sveden na problem nalaženja sume u datom podsegmentu. Ovaj upit možemo da odgovorimo sa vremenskom složenošću $O(log\ n)$ gdje je n broj čvorova u stablu korišćenjem segmentnog stabla.

Za upite prvog tipa potreno je da promijenimo vrijednosti u nizu na dva odgovarajuća indeksa koji predstavljaju početak i kraj odgovarajućeg segmenta, I da ažuriramo segmentno stablo. I ova operacija se može izvršiti sa vremenskom složenošću $O(log\ n)$.

Međutim ukupna vremenska složenost upita tipa 2 je $O(number\_of\_children * log\ n)$, jer je potrebno da za svakog potomka čvora pronađemo sumu njegovog podstabla. Tako da imamo ukupnu složenost $O(n * q * log\ n)$. Ovo rješenje je bilo dovoljno dobro za 70% poena.

Posmatrajmo broj čvorova koji imaju više od sqrt(n) potomaka.
Svaki čvor u stablu je sin najviše jednog čvora, pa je maksimalan broj čvorova koji imaju više od sqrt(n) potomaka sigurno manji od sqrt(n). Upite tipa 1 za čvorove koji imaju manje od sqrt(n) potomaka ćemo odgovarati na isti način kako smo i do sada radili.

Upite za čvorove sa više od sqrt(n) potomaka ćemo odgovarati sa složenošću $O(1)$. Da bi to postigli, prilikom dodavanja čvora u stablo treba da obiđemo sve čvorove sa više od sqrt(n) potomaka koji se u stablu nalaze na putanji između čvora sa indeksom 1 i čvora koji smo upravo dodali. Neka je X čvor koji dodajemo u stablo. I neka je Y_k, k-ti čvor na putanji od čvora X do korijena stabla takav da Y_k ima više od sqrt(n) potomaka. I neka je Z_k potomak čvora Y_k, koji se nalazi na putanji od čvora X do čvora Y_k. Primijetimo da je Z_k jedinstveno određen. Treba da dodamo vrijednost X sumi podstabla sa korijenom u Z_k, i ako je potrebno ažuriramo rješenje za Y_k da predstavlja sumu podstabla sa korijenom u Z_k. Tako da je ukupna vremenska složenost operacije tipa 1 $O(log\ n$ + sqrt(n)).