Zadatke, test primjere i rješenja pripremili: Alen Rakipović, Frane Kurtović, Ivan Mandura, Ivo Sluganović, Tomislav Gudlek, Goran Žužić i Ante Đerek. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima koji nužno ne odgovaraju u svim detaljima ovdje opisanim algoritmima.

SKOKOVI

Predložio: Frane Kurtović
Potrebno znanje: osnovne aritmetičke operacije, naredba if

Ukupna ocjena se sastoji od dvije komponente koje je potrebno zasebno izračunati i zbrojiti.

- Prva komponenta je zbroj tri srednje ocjene od njih pet. Moguće ju je izračunati tako da se zbroji svih pet ocjena i od toga oduzme najveća i najmanja ocjena. Najveću ocjenu je moguće odrediti uzastopnim primjenjivanjem naredbe *if*, ili pak korištenjem funkcije *max* u jeziku C++. Na sličan način je moguće naći i najmanji broj.
- U drugoj komponenti skijaš osvaja 60 bodova ako je skočio točno **K** metara, a za svaki metar više od toga osvaja 2 dodatna boda, dok za svaki metar manje gubi 2 boda. Time je zapravo izrečena jednostavna matematička formula koju je moguće zapisati kao *60 + (D - K) * 2*. Ako je **D** veći od **K** tada je razlika pozitivna, a inače je negativna.

Za one koji žele više

Prvu komponentu je moguće izračunati tako da se brojevi učitaju u niz duljine 5. Nakon toga je moguće vrlo jednostavno sortirati niz korištenjem funkcije *sort* iz standardne biblioteke *algorithm* u programskom jeziku C++. Ta funkcija sortira niz uzlazno, što znači da je prvi element najmanji, a zadnji najveći, što su upravo one dvije vrijednosti koje su potrebne za izračunati prvu komponentu.

NAZAD

Predložio: Ivo Sluganović
Potrebno znanje: razlaganje broja na znamenke

U zadatku je objašnjen očekivan postupak računanja produžetka broja, a ograničenja zadatka su takva da nije potrebno paziti na prelijevanje ukoliko se koriste 32-bitni cijeli brojevi. Kako je produžetak broja uvijek veći od početnog broja, dovoljno je jednostavno isprobati sve prirodne brojeve između 1 i **B** (koji je veći) te provjeriti za koji od isprobanih brojeva bi sljedeći broj u nizu bio **B**.

Računanje produžetka najbolje je odvojiti u zasebnu funkciju kako bismo jednostavnije provjerili ispravnost ovog dijela koda.

- Funkcija *int produzetak(int n)* ulazni će broj spremiti u pomoćnu varijablu *tmp*, a zatim "izvaditi" jednu po jednu znamenku broja *tmp* tako da u svakom koraku nađemo posljednju znamenku broja kao ostatak cjelobrojnog dijeljenja broja tmp sa 10, a zatim odbacimo posljednju znamenku broja *tmp* cjelobrojnim dijeljenjem broja sa 10.

Za one koji žele više

Zamislite niz prirodnih brojeva gdje je svaki sljedeći broj jednak produžetku posljednjeg broja u nizu (a ne zbroja, kao u zadatku). Za svaki početni broj manji od 1000, nizovi će se u nekom trenutku "spojiti", nakon čega će biti isti. Provjerite to!

AUTOCESTE

Predložio: Ivan Mandura
Potrebno znanje: polja, stringovi, sortiranje

Zadatak se može riješiti na nekoliko načina, a uvelike mogu pomoći dinamičke strukture podataka poput vector i map u C++-u. Ovdje ćemo opisati jedno rješenje koje koristi samo polja. U prvoj fazi rješenja cilj je za svaki automobil pronaći sve njegove prolaske. Registraciju automobila ćemo spremiti u polje *registracija*, a za svaki automobil, prolaske tog automobila ćemo spremiti u dvodimenzionalna polja *kucica* i *vrijeme*, sam broj prolazaka pamtimo pomoću polja *broj_prolaza*. Potrebne dimenzije polja možemo izračunati na temelju ograničenja iz zadatka.

```
int registracija[100]
int broj_prolaza[100];
string kucica[100][100];
string vrijeme[100][100];
```

Sada, prilikom čitanja ulaza, za svaki prolaz najprije nađemo indeks trenutne registracije u polju *registracija*, odnosno dodamo novu registraciju ako trenutnu registraciju nismo vidjeli prije. Nakon toga dodajemo trenutno vrijeme i ime kućice u odgovarajuća polja *kucica* i *vrijeme* na nađenom indeksu.

U drugom dijelu zadatka trebamo izvršiti sortiranje prolaza za svaki automobil. Za sortiranje možemo koristiti jednostavni algoritam kao na primjer selection sort. Prilikom sortiranja prolaza trebamo paziti da zamjenjujemo i ime kućice i vrijeme prolaza. Sortiranje po registraciji se radi slično.

ISTRAGA

Predložio: Ivo Sluganović
Potrebno znanje: razlaganje broja na znamenke

U zadatku je objašnjen očekivan postupak računanja, a ograničenja zadatka su takva da nije potrebno paziti na prelijevanje ukoliko se koriste 32-bitni cijeli brojevi.

Stoga, najjednostavnije je napisati funkciju koja računa *produžetak* broja, a zatim jednom petljom izračunati traženih **N** brojeva. U petlji pamtimo zadnja dva člana niza i iz njih računamo sljedeći član u nizu korištenjem funkcije *produzetak()*. Nakon računanja novog člana, potrebno je promijeniti iznose "posljednja dva člana niza", što činimo postavljanjem:

```
a = b;
b = novi;
```

Funkcija *int produzetak(int n)* ulazni će broj spremiti u pomoćnu varijablu *tmp*, a zatim "izvaditi" jednu po jednu znamenku broja *tmp* tako da u svakom koraku nađemo posljednju znamenku broja kao ostatak cjelobrojnog dijeljenja broja tmp sa 10, a zatim odbacimo posljednju znamenku broja *tmp* cjelobrojnim dijeljenjem broja sa 10.

Za one koji žele više

Zamislite niz prirodnih brojeva gdje je svaki sljedeći broj jednak produžetku posljednjeg broja u nizu (a ne zbroja, kao u zadatku). Za svaki početni broj manji od 1000, nizovi će se u nekom trenutku "spojiti", nakon čega će biti isti. Provjerite to!

RUB

Predložio: Frane Kurtović
Potrebno znanje: dvodimenzionalna polja, osnovne petlje

Zbog relativno niskih ograničenja u ovome zadatku (R, S ≤ 30) moguće je provjeriti svaki postojeći kandidatni pravokutnik. Koliko uopće ima takvih pravokutnika? Pravokutnik je određen svojim gornjim lijevim kutom (R×S mogućnosti) te svojim donjim desnim kutom (R×S mogućnosti). Nisu sve od mogućnosti moguće (npr. ako gornji desni kut ispadne ispod gornjeg lijevog kuta), al nam gornji izbor omogućuje da u 4 petlje i nekoliko uvjeta pobrojimo sve kandidatne pravokutnike.

Nakon što smo fiksirali neki pravokutnik, potrebno je provjeriti jesu li na njegovom rubu sve pločice jednake vrste. Najlakši način za to napraviti je provjeriti jesu li svi pločice na rubu jednake pločici u gornjem lijevom rubu pravokutnika što nas dovodi do konačne implementacije s ukupno 5 ugniježđenih petlji. Za detalje pogledajte priložene kodove rješenja.

Za one koji žele više

Kako biste zadatak riješili koristeći samo 4 ugniježđene petlje (odnosno u složenosti $O(R^2S^2)$)?

STABLO

Predložio: Alen Rakipović
Potrebno znanje: nizovi, osnove teorije grafova

Zadatak nas traži da spremimo zadano stablo te ga obiđemo "po nivoima" (gdje je nivo Perice označen s 1, nivo njegovih izravnih podređenih s 2, i tako dalje). Ovom problemu se može dosta jednostavno pristupiti algoritmom gdje ćemo za svaku razinu L pronaći sve vrhove koji se nalaze na toj razini (i sve takve vrhove ćemo pospremiti u niz E_L):

1. stavi u niz E_1 vrh s oznakom 1 (direktora Pericu)
2. ponavljaj od L = 1, 2, … dok god nismo obišli sve vrhove:
    a. za sve vrhove x spremljene u niz E_L
        i. spremi desnu ruku vrha x (ako postoji) u E_{L+1}
        ii. spremi lijevu ruku vrha x (ako postoji) u E_{L+1}
3. ispiši sve vrhove onim redoslijedom koji smo ih pronalazili

Lako se uoči kako ovaj algoritam u točnom redoslijedu ispisuje sve vrhove zadanog stabla.

Za one koji žele više

Zadatak ustvari traži da obiđete zadani graf algoritmom obilaska-u-širinu (BFS). Vidite li zašto? Kako biste provjerili za općeniti graf čini li korektno stablo opisano u zadatku? Kako biste riješili ovaj zadak u O(1) dodatne memorije (nakon učitavanja originalnog grafa)?