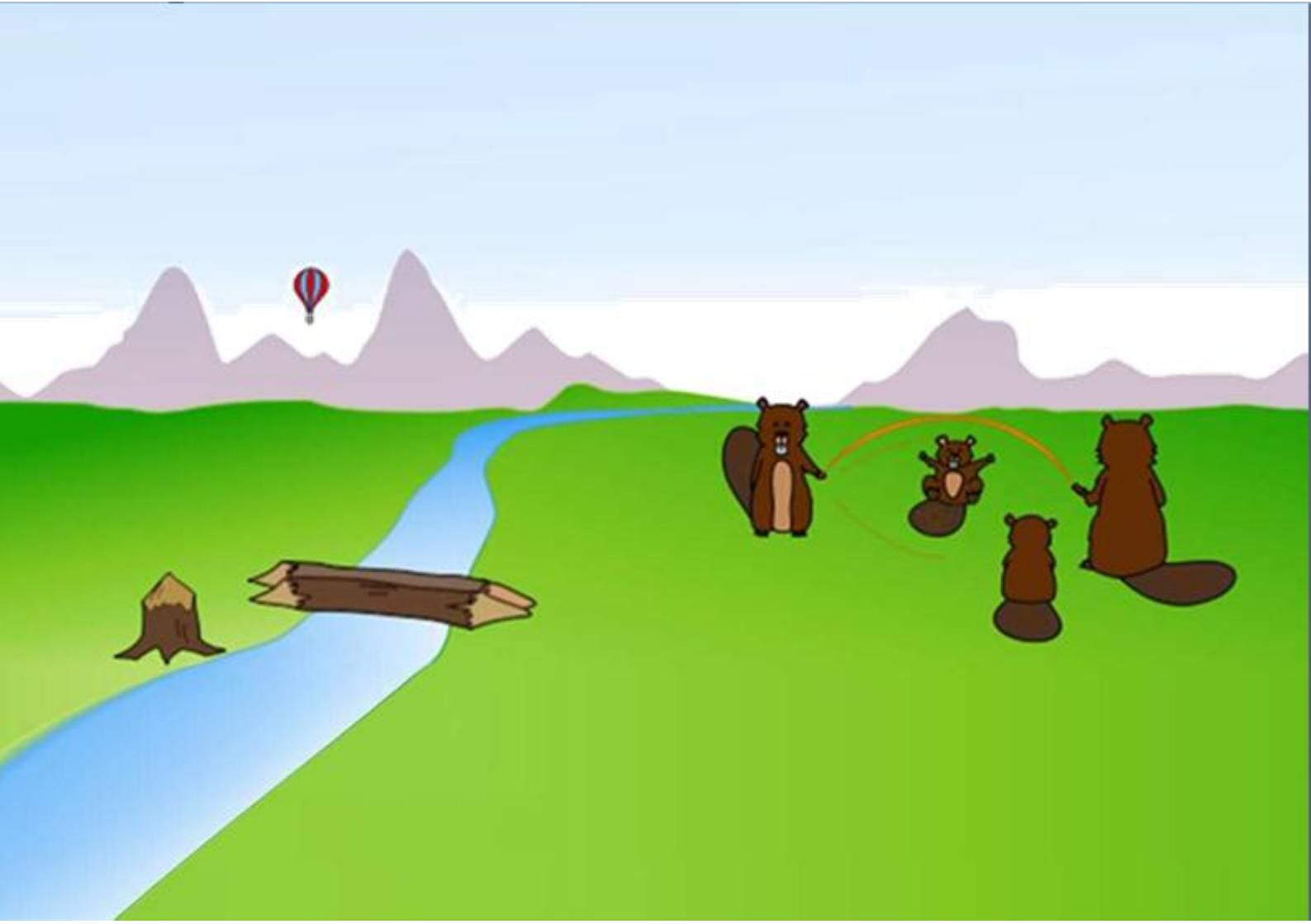


Dabar_2016

Zadaci i rešenja za učenike srenjih škola i gimnazija – školsko takmičenje

Izbor zadataka za tekmičenje i prevod: Programski odbor takmičenja,

Milan Rajković (predsednik programskog odbora)
Svetlana Jakšić (član programskog odbora)
Milan Lukić (član programskog odbora)
Bojan Tomić (član programskog odbora)
Bojan Milosavljević (član programskog odbora)
Dragan Krstić (član programskog odbora)
Saša Jevtić (član programskog odbora)

Tehnična podrška: Milutin Spasić i Branko Dolić

O takmičenju i priručniku

Draga deco takmičari i poštovane kolege mentori,

U toku je još jedna takmičarska Dabar godina u kojoj imate priliku da se kroz učešće na Međunarodnom takmičenju Dabar sretnete sa novim informatičkim izazovima. U ovom priručniku smo predstavili zadatake koje ste rešavali na školskom nivou takmičenja za 2016. godinu.

Želja nam je da priručnik pomogne nastavnicima i učenicima u bavljenju temama i intelektualnim problemima koji su predstavljeni u zadacima. Kako deca vole da se takmiče i vole da razmišljaju, naš posao je da ih i tokom godine podstičemo da razijaju takmičarski duh i radoznalost.

Priručnik je predviđen za učenike, ali ne samo kao priprema za buduća takmičenja, već i kao deo riznice Dabar intelektualnih problema, koja se iz godine u godinu uvećava. Pripručnik su pripremili organizatori takmičenja, kao nešto na šta smo ponosni ;). U takmičenje je uloženo mnogo rada i energije, tako da nas posebno raduje što takmičenje postaje sve masovnije i popularnije, ne samo u našoj zemlji već i u svetu.

Trenutno izdanje prizučnika je "privremeno". Nakon narednog nivoa takmičenja, našoj riznica zadataka ćemo dodati nove. No, čak i oni će biti veoma brzo odrađeni od strane onih koji vole takve zadatke. Šta onda?

Pozivamo vas da pratite naše aktivnosti i da sa zajedno sa nama radujete novim zadacima.

Želimo vam da uživate u rešavanju zadataka!

Programski odbor takmičenja

Alonsov računar obrađuje informacije na vrlo poseban način, koristeći veoma malo operacija:

- (max x_1 x_2 … x_n)uzeće maksimum svih vrednostix$_1$ x_2 … x_n
- (min x_1 x_2 … x_n)uzeće minimum svih vrednostix$_1$ x_2 … x_n
- (+ x_1 x_2 … x_n)izračunaće kaox$_1$ + x_2 + … + x_n
- (· x_1 x_2 … x_n)izračunaće kaox$_1$ · x_2 · … · x_n

On može ugnežđavati ove operacije u izrazima, npr. (+ (· 2 3) (+ 1 2)) gives the value 9.

Koja je vrednost izraza:

+ (max (min 3 9 2) (· (max 0 4) (min 0 4))) (min (max 3 6) (max 5 7 2)))

A) 5

B) 8

C) 13

D) 0

Obrazloženje

Izraz se može izračunati od iznutra ka spolja (podvučeni delovi se izračunavaju u sledećem redu):

(+ (max <u>(min 3 9 2)</u> (· <u>(max 0 4)</u><u>(min 0 4)</u>)) (min <u>(max 3 6)</u><u>(max 5 7 2)</u>))

⇒(+ (max 2 <u>(· 4 0)</u>) <u>(min 6 7)</u>)

⇒(+ <u>(max 2 0)</u> 6)

⇒<u>(+ 2 6)</u>

8

Informatička pozadina:

Ovo pitanje bavi se funkcionalnim programiranjem i sposobnošću da se ugnežđavaju izrazi. Ideja ugnežđavanja se pretežno koristi u tabelarnim proračunima: sakupljanje podataka primenom jedne funkcije za generisanje podataka (npr. sumiranjem svih brojeva), koje prati primena druge funkcije koja generiše neke druge podatke (kao što je maksimalna vrednost svih tih suma).

Ovaj model računarske obrade koristi se na više različitih načina: tri takva programska jezika su Lisp, Scheme i Racket, a ovaj koncept takođe se koristi i u savremenijim programskim jezicima,

kao što su lambda funkcije u jeziku C# ili funkcije za preslikavanje (map) i izdvajanje (filter) u jeziku Python.

Na teorijskom nivou, ovo je poseban oblik prefiksne notacije u kome se operator postavlja pre operanada. Iako se ovo koristi samo za regularne operatore kao što su +, -, · ili ÷, prefiksna notacija se takođe može koristiti i za operatore koji rade sa listom brojeva, kao što su min ili max u ovom zadatku, ali to zahteva korišćenje zagrada, za razliku od obične prefiksne notacije, koja to ne zahteva.

Ključne reči:
Ugnežđavanje, funkcionalno programiranje.

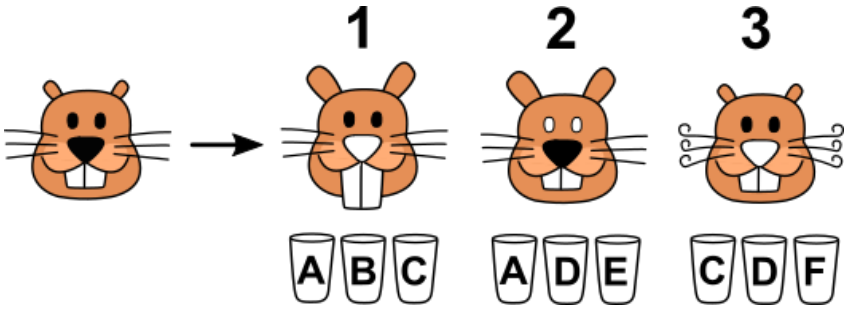
Magični napici

1. i 2. razred SŠ (Dabar)
3. i 4. razred SŠ (Stariji dabar)

Dabar Bole je otkrio pet vrsta čarobnih napitaka sa sledećim efektima:
• jedan napitak pravi duže uši;
• drugi napitak pravi duže zube;
• sledeći čini brkovi kovrdžavim;
• sledeći boji nos u belu boju;
• sledeći boji oči u belu boju.

Bole je svaki napitak stavio u posebnu čašu i u jednu čašu je stavio čistu vodu koja nema čarobno dejstvo. Čaše je obeležio slovima A,B, C, D, E i F. Međutim, Bole je zaboravio da napiše u kojoj čaši je koji napitak.

Tada je Bole uradio tri eksperimenta na različitim dabrovima:
• Eksperiment 1: kada popije napitke iz čaša A, B i C dobija se dabar kao na slici 1,
• Eksperiment 2: kada popije napitke iz čaša A, D i E dobija se dabar kao na slici 2,
• Eksperimant 3: kada popije napitke iz čaša C, D i F dobija se dabara kao na slici 3



Pitanje: koja kombinacija napitaka iz čaše, će dovesti do jedne promene na dabru?

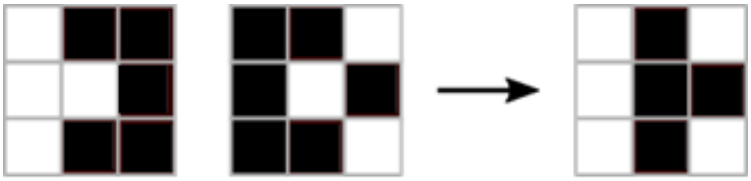
A) kada popije napitke iz čaša A i B
B) kada popije napitke iz čaša C i D
C) kada popije napitke iz čaša E i F
D) kada popije napitke iz čaša A i E

Obrazloženje

Tačan odgovor**:** B) kada popije napitke iz čaša C i D

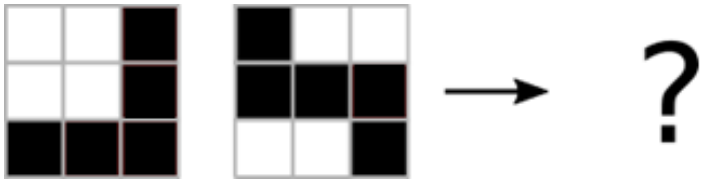
Informatička pozadina: Ovo se može uraditi pomoću logičkog rasuđivanja. Logika igra važnu ulogu u informatičkoj nauci. Najmanja jedinica za količinu informacija je bit, koji ima vrednost 1 (true) ili 0 (false). Sve ostale informacije u računaru se čuvaju koristeći specifične kombinacije bita. Ovaj problem takođe istražuje osnovne teorije skupova.

Kombinacijom kartica A i B dobija se kartica C.



Kartica A Kartica B Kartica C

Pitanje: koliko će crnih polja biti u kartici koja će se dobiti kombinacijom kartice D i kartice E (prikazano na slici ispod)?



Kartica D Kartica E

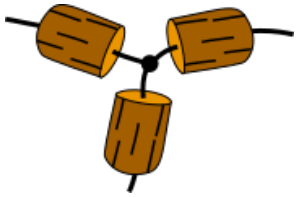
A) 1
B) 2
C) 3
D) 4

Obrazloženje

Tačan odgovor: C) 3

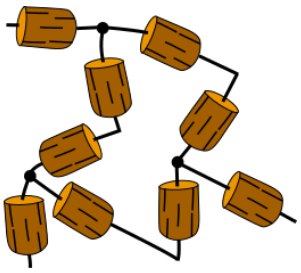
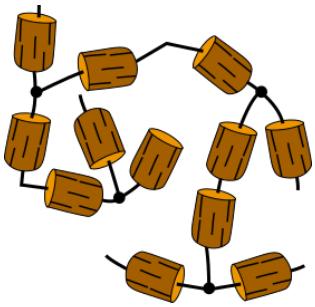
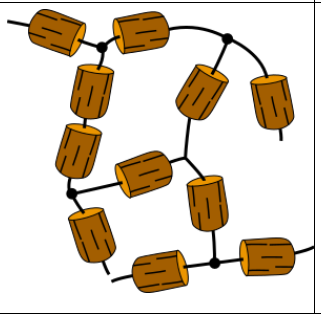
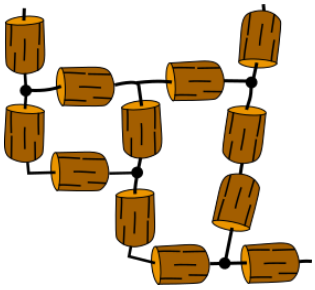
Informatička pozadina: Bulovi izrazi. Bela polja možemo tumačiti kao 1 (true) dok crna polja možemo tumačiti kao 0 (false). Pored osnovnih operacija (i, ili, ne) postoje i izvedene operacije. Jedna od njih je eksplicitno ili (XOR) i daje rezultat 1 ako i samo ako je tačno jedna promenljiva jednaka 1.

Dabar Karlo je našao punu kutiju ispunjenu identičnim igračkama, kao na slici ispod.



Dabar Karlo želi da napravi velike igračke od igračaka koje je pronašao.

Pitanje: Koju igračku dabar Karlo na može da napravi?

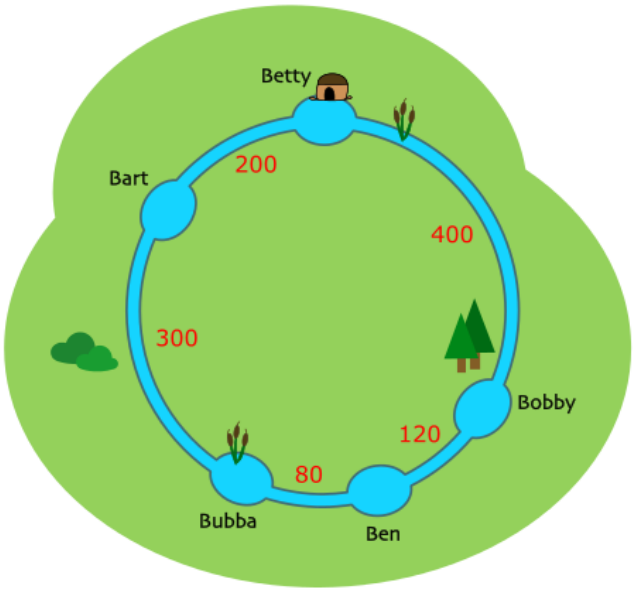
A	B	C	D
			

Obrazloženje

Tačan odgovor: C

Informatička pozadina: Proces izgradnje složenog objekta od jednostavnijih objekata se naziva kompozicija. U stvarnom životu, složeni objekti se često grade od manjih, jednostavnijih objekata. Na primer, automobil je napravljen od metalne kontrukcije, motora, točkova, prenosa snage, volana i ostalih brojnih delova.

Pet dabrova, veoma srećno, žive u jednom kružnom kanalu. Odlučili su da se sastanu kod jednog od njih. Oni žele da naprave sastanak kod onog dabra do kog će ukupna dužina plivanja koju će preći biti najmanja.



Pitanje: Kod kog dabra će se održati sastanak?

a. Bobby
b. Ben
c. Bubba
d. Bart

Obrazloženje

Tačno rešenje: c. Bubba

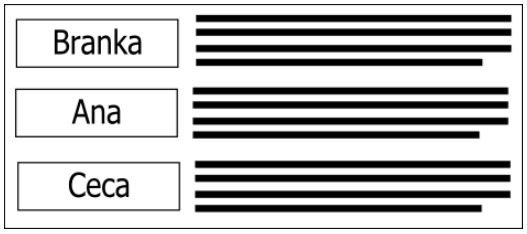
Informatička pozadina: Pronalaženje najkraćeg puta (rastojanja) je algoritamski problem koji se javlja u mnogim situacijama.

Kružna struktura nam daje više mogućnosti dolaska do cilja ali treba naći najbolje rešenje (ušteda vremena, memorije...).

Postoje dva softverska paketa za korišćenje e-pošte: P-Mail i K-Mail.

Kada prosleđuje e-poštu drugoj osobi, P-Mail uvek dodaje sadržaj nove e-pošte na početku postojeće prepiske, dok K-Mail uvek dodaje novu e-poštu na kraj prepiske.

4 prijatelja šalju jedni drugima e-poštu. Ana i Branka koriste samo P-Mail. Ceca i Dijana ponekad koriste P-Mail, a ponekad i K-Mail.

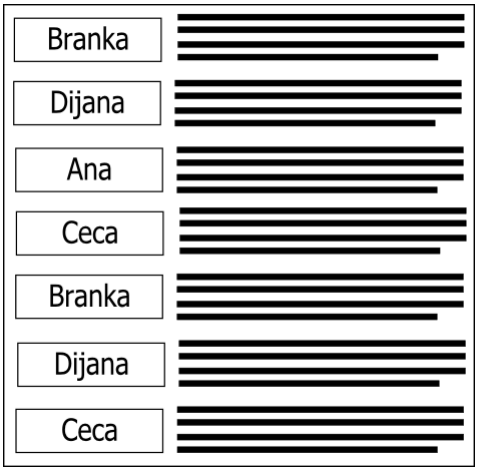


Pretpostavimo da je Ana prvi pošiljalac i da šalje e-poštu Ceci. Ceca koristi K-Mail da prosledi e-poštu Branki. Konačno, Branka prosleđuje e-poštu Dijani.

Dobijena prepiska e-pošte onda izgleda kao na slici prikazanoj desno.

Pitanje:

Sledeća slika prikazuje još jednu prepisku e-pošte. Nije jasno ko je poslao prvu e-poruku. Tabela desno prikazuje ko je koristio koji programski paket za korišćenje e-pošte.



Korisnik	Koristi softver za e-poštu
Ana	P-mail
Branka	P-mail
Ceca	P-mail, K-mail
Dijana	P-mail, K-mail

Ko **nije** mogao biti pošiljalac prve e-poruke?

A) Ana

B) Branka

C) Ceca

D)Bilo ko je mogao poslati prvu e-poruku

Obrazloženje

Tačan odgovor je pod A) Ana.

Ana nije mogla biti prvi pošiljalac. Ako bi Ana bila prvi pošiljalac, Brankina e-pošta ne bi se pojavila ispod Anine.

Ako je Branka bila prvi pošiljalac, onda se prosleđene e-poruke od Ane ili Branke mogu pojaviti iznad prve e-poruke. E-poruke prosleđene od Cece ili Dijane mogu biti iznad ili ispod prethodne poruke. Slično je i ako bi Ceca ili Dijana bile prvi pošiljalac.

Shodno tome, svaka od tri preostale osobe bi mogla biti pošiljalac prve e-poruke.

Informatička pozadina:

Razumevanjem konačnog stanja procesa i skupa pravila koja upravljaju procesom, može se koristiti logičko razmišljanje unazad i zaključiti početno stanje. U ovom slučaju, mogu se pronaći mogući pošiljaoci prve e-poruke, kao i utvrditi koji je softver za rad sa e-poštom korišćen.

E-pošta i forumi usvajaju oba stila slanja: slanje na početku i slanje na kraju. Softver za rad sa e-poštom obično ima podrazumevani stil slanja, iako se obično može promeniti od strane korisnika. Kod foruma, iako se koriste oba stila, svaka Internet zajednica razlikuje se po tome koji je stil slanja podesan i prihvatljiv.

Ključne reči:
Stil slanja e-pošte, Poslata nova e-pošta na početku, Poslata nova e-pošta na kraju.

Igra šibicama

1. i 2. razred SŠ (Dabar)
3. i 4. razred SŠ (Stariji dabar)

Boban sreće prijatelja i želi da igraju igru sa šibicama. On objašnjava igru svom prijatelju: "Postoji 13 šibica u redu. Igrač 1 započinje igru uklanjanjem 1,2 ili 3 šibice. Onda je na potezu Igrač 2, koji takođe uklanja 1, 2 ili 3 šibice. Onda se naizmenično smenjuju ponovo Igrač 1, pa Igrač 2 itd. Igrač koji ukloni poslednju šibicu dobija igru.

Boban započinje igru.

Pomoć: Ako ostane 4 šibica Boban ne može uzeti poslednju šibicu. On mora da izbegne ovakvu situaciju!

Koliko šibica treba da ukloni Boban u svom prvom potezu da bi dobio igru?

A)Jednu šibicu

B)Dve šibice

C)Tri šibice

D)Nije bitno koliko šibica uzme

Obrazloženje

Tačan odgovor A) Jednu šibicu.

Boban mora prvo ukloniti jednu šibicu. Ostaje 12 šibica.

U sledećim potezima Boban uklanja šibice na takav način da broj preostalih šibica bude umnožak od 4. Prema ovoj strategiji njegov prijatelj će završiti igru sa 4 šibice i Boban može dobiti igru.

Informatička pozadina:

Ovo je klasičan primer strategijske igre za dva igrača sa naizmeničnimređanjem poteza. Pposle svakog poteza računar analizira različite moguće poteze i izračunava najveću verovatnoću da dobije igru. Onda računar izvršava svoj potez i ponovo započinje analizu.

Postoji tačan algoritam za dobijanje ove igre. Za druge igre, koje nemaju tačan algoritam za pobeđivanje, ili kada je tačno određeni algoritam za dobijanje igre spor, primenjuju se heurističke metode. 1997. godine, prvi put u istoriji, svetski šampion u šahu je poražen od strane računara koji koristi heuristiku.

Ključne reči:

Igre, dopuna (komplement) po modulu, stablo odlučivanja.

Dabar Bobi dobio je kuvar za Božić. On mora da programira recept korak po korak koristeći instrukcije. Svaka instrukcija počinje brojem. Ako su potrebni neki sastojci za zadatak, odgovarajuće slovo za dati sastojak mora se navesti u zagradi, posle broja. Svaka instrukcija mora biti u posebnom redu.

Npr.: instrukcija "*Pomešaj brašno i ulje*". "*Kuvaj potrebno vreme*" mora se programirati na sledeći način:

4 (B, SU)
2

Instrukcije:
1. Dodaj
2. Kuvaj potrebno vreme
3. Isprži
4. Pomešaj
5. Isključi

Sastojci:	
CL. Crni luk	KP. Kisela pavlaka
P. Paprika	B. Brašno
V. Voda	SU. Suncokretovo ulje
PI. Piletina	Z. Začini

Bobi će kuvati pileći paprikaš po sledećem receptu:

- Isprži crni luk i suncokretovo ulje.
- Dodaj papriku, vodu i piletinu.
- Kuvaj potrebno vreme.
- Pomešaj kiselu pavlaku i brašno zajedno u činiji.
- Dodaj pomešanu kiselu pavlaku i brašno u tanjir za prženje.
- Dodaj začine.
- Kuvaj potrebno vreme.
- Isključi.

Koji će od sledećih skupova instrukcija skuvati pileći paprikaš?

A)	B)	C)	D)
2	3 (SU, CL)	3 (SU, CL)	3 (SU, CL)
3(SU, CL)	1(P, V, PI)	1(P,V, PI)	1(P, V, PI)
2	2	2	2
4(KP, B)	4(KP, PI)	4(KP, B)	4(KP, PI)
2	1 (KP, B)	1 (KP, B)	1
5	2	1 (Z)	5
	5	2	2
		5	5

Obrazloženje

Tačan odgovor je pod C).

Odgovor pod A) se odbacuje, jer započinje kuvanjem pre nego što su dodati bilo koji sastojci.

U odgovoru pod B), u četvrtom redu program kaže 4(KP, PI). Umesto piletine, potrebno je pomešati brašno, i shodno tome odgovor je netačan.

U šestom redu odgovora pod D) isključuje se kuvanje, posle čega sledi nemoguća akcija nastavka kuvanja.

Informatička pozadina:

Strukturno programiranje koristi tri algoritamske konstrukcije: sekvencu, selekciju i iteraciju. Ovaj zadatak je primer sekvence. Svaki ppodprogram ili instrukcija izvršavaju se jedna za drugom, po redu navođenja.

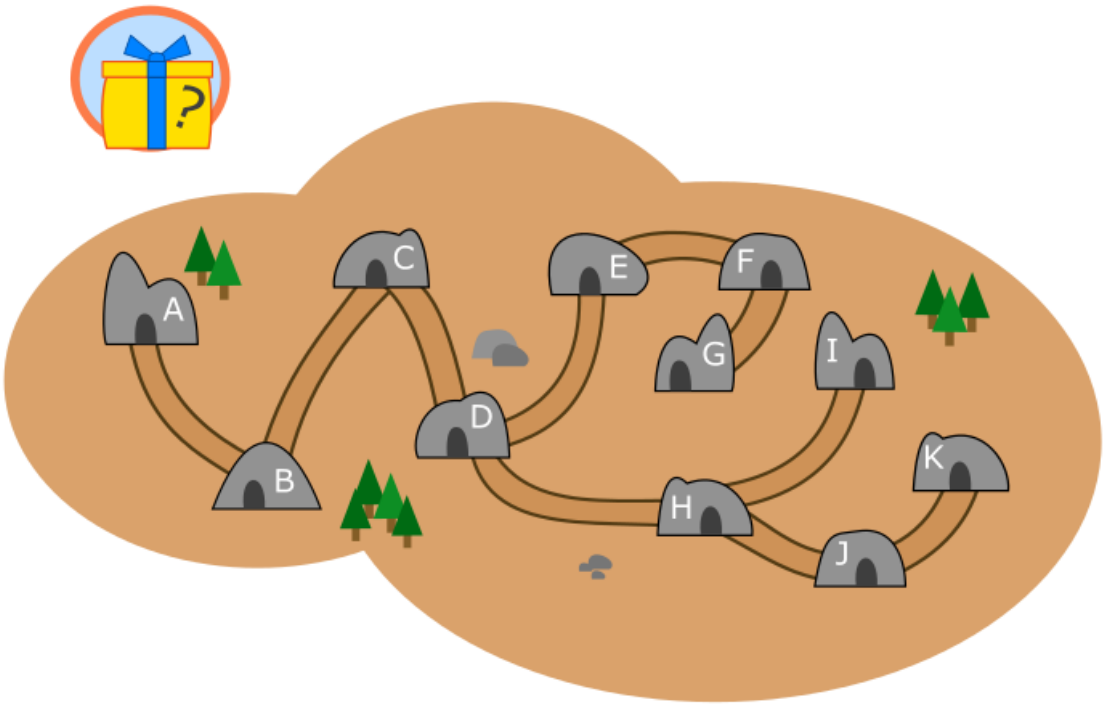
U programiranju, možemo pozvati podprograme korišćenjem parametara kojima se prosleđuju podaci u podprogram. U ovom zadatku, procedure “Dodaj”, “Pomešaj” i “Isprži”koriste različite parametre, u zavisbnosti od toga koji su sastojci potrebni. Ovo je veoma korisno u programiranju, jer se isti podprogram može koristiti ponovo i ponovo sa različitim podacima.

Ključne reči:
Sekvenca, podprogram, parametar, instrukcija.

Igra u pećini

1. i 2. razred SŠ (Dabar)
3. i 4. razred SŠ (Stariji dabar)

U Dabrogradu postoji kvart u kome se nalazi nekoliko pećina. Neke pećine su povezane prolazima; između dve pećine može postojati samo jedan prolaz, kao što je prikazano na slici.



Dabrovi Bob i Partik žive u Dabrogradu i igraju se igre skrivalice. Bob je sakrio poklon u jednoj od pećina a Patrik želi da pronađe poklon.

Kako se igra: Partik uzima mapu pećina i jedino što može da pita je: Da li je poklon u pećini koja nosi slovo X (X je jedno od slova kojima su obeležene pećine)? Bob može da odgovori sa "Da", ukoliko je Partik pogodio pećinu. Ukoliko je Partik pogrešio, Bob odgovara sa slovom koje predstavlja najbližu, prolazom spojenu, pećinu koja se nalazi na putu ka pećini u kojoj je poklon.

Pitanje: Koji je najmanji mogući broj pitanja, da bi Partika sa sigurnošću pronašao poklon?

A. 3

B. 4

C. 5

D. 6

Obrazloženje

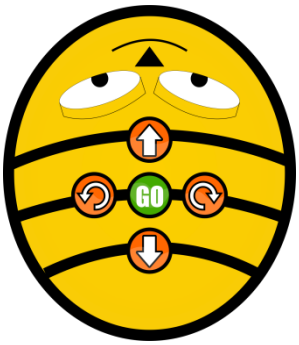
Tačan odgovor je pod A. 3

Informatička pozadina: U pitanju je binarna pretraga. U ovom pitanju pećine su čvorovi stabla. Najveća prednost binarnog stabla pretrage u odnosu na ostale strukture podataka je da algoritmi sortiranja i algoritmi pretrage kao npr. pretraga u dubinu (in-order) mogu biti veoma efikasni.

Vrati se nazad

1. i 2. razred SŠ (Dabar)
3. i 4. razred SŠ (Stariji dabar)

Robot pčela ima na svojim leđima četiri tastera sa strelicama. Pčela se pomera po podu sa kvadratnim poljima prema skevenci koju zadaš pritiskom na sledeće komandne tastere:

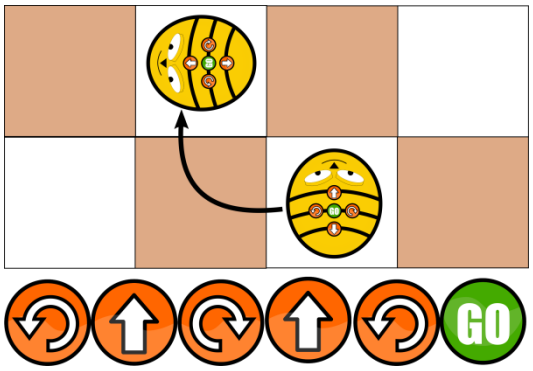


 pomeri se napred u sledeće polje

 skreni levo 90° u istom polju

 skreni desno 90° u istom polju

 pomeri se unazad u polje iza pčele



Taster na leđima pčele započinje izvršavanje unete sekvence. Primer pritisaka tastera i odgovarajućeg kretanja pčele prikazan je na slici.

Pčela zapamti unetu sekvencu, tako da ponovni pritisak tastera GO dovodi do istog kretanja. Primeti da bez obzira na sekvencu koju uneseš, ukoliko uzastopno više puta pritisneš taster GO pčela će se ili vratiti na početnu poziciju sa istom orijentacijom, ili se nikada neće vratiti.

Broj je *broj za zujanje* ako se može uneti sekvenca kojom se upravlja kretanje pčele tako da posle pritiska tastera GO taj broj puta pčela se vraća po prvi put u svoju početnu poziciju sa istom orijentacijom.

Koji je najveći mogući broj za zujanje?

A) 2

B) 4

C) 8

D) Ne postoji najveći mogući broj za zujanje

Obrazloženje

Ispravan odgovor je B) 4.

Prvo dokazujemo da je jedino od značaja pravac (orijentacija) pčele. Pretpostavimo da smo napisali sekvencu koja vraća pčelu na njenu početnu poziciju. Pretpostavimo takođe da se prvi put pčela vraća u početnu poziciju posle ptitiska tastera GO taj broj puta. Ako je mesto gde se nalazi sada pčela početno, onda smo uradili traženo. Inače je naše rastojanje od početnog polja ne-nula. Budući da je ovaj izbor bio najmanji, ako pritisnemo ponovo taster GO isti broj puta, onda još povećavamo naše rastojanje i možemo dalje nastaviti ovaj proces na ovaj način. Ni u jednom drugom trenutku pčela se ne vraća u svoju početnu poziciju,

osim ako ne pritisnemo taster GO umnožak ovog minimalnog broja puta (u suprotnom bi postojao manji broj koji bi vratio pčelu na njenu početnu poziciju). Tako će rastojanje od početnog polja uvek biti ne-nula, što protivreči činjenici koju smo pretpostavili da se naša pčela vratila u početno polje.

Dakle, ključni deo programa koji određuje broj za zujanje je orijentacija (i nezavisna je od ukupnog pređenog rastojanja, jer sledi da ono mora biti 0).

Možemo razvrstati sve moguće sekvence za upravljanje kretanjem pčele u tri različite grupe prema promeni pravca pčele:

Jedna iteracija sekvence ne menja pravac. Da bi onda pčela došla nazad, mora se vratiti posle 1 pritiska tastera GO. U ovom slučaju, broj za zujanje je 1.

Jedna iteracija sekvence menja pravac 180° (okreće pčelu na glavu). Sledeće izvršavanje programa će okrenuti pčelu u početnom pravcu i tako je mora postaviti u početno polje. U ovom slučaju, broj za zujanje je 2.

Jedna iteracija sekvence menja pravac 90° levo ili desno. Posle 3 dodatna pritiska na taster GO pčela će se vratiti u početni pravac i tako morati da se postavi u početnu poziciju. U ovom slučaju, broj za zujanje je 4.

Budući da ovo obuhvata sve moguće slučajeve, maksimalni broj za zujanje je 4.

Informatička pozadina:

Ima mnogo načina za predstavljanje ovakve situacije u informatici. Pčela ovde predstavlja računar, a sekvenca program koji računar izvršava. Taster GO simulira cikličnu strukturu petljena ovom programskom jeziku koji se predstavlja tasterima sa strelicama.

Brojevi za zujanje su ovde objekat koji se želi optimizovati. Mnogo puta u informatici se traži projektovanje rešenja problema za scenarije najboljeg ili najgoreg slučaja. Ovde se traži takav najveći broj za zujanje, odnosno najgori broj slučaja pritiska tastera GO da bi pčela najbrže stigla u početno polje. Ovo rešenjepredstavlja brzi algoritam kojim se može odrediti da li će se pčela ikad vratiti u svoje početno stanje ili ne. Pritisni taster GO 4 puta. Ako ikada uočiš da se pčela vraća u svoje početno stanje, vrati “DA”, inače vrati “NE”.

U informatici je uopšteno govoreći teško pitanje da li se neki algoritam završava. To se zove Problem zaustavljanja. U ovom slučaju, može se brzo odrediti ponašanje algoritma na osnovu nekih fundamentalnih matematičkih principa, ali u opštem slučaju je ovo teško (ili čak nemoguće) uraditi.

Pozicija i lokacija pčele se takođe modeluje u okviru stanja. Stanje ovde ima dva atributa, i za data ta dva atributa može se tačno odrediti gde se pčela nalazi na tabli sa poljima.

Ključne reči:

Kretanje robota, pronalaženje konačnog stanja, petlja, optimizacija.

Rekurzivno crtanje

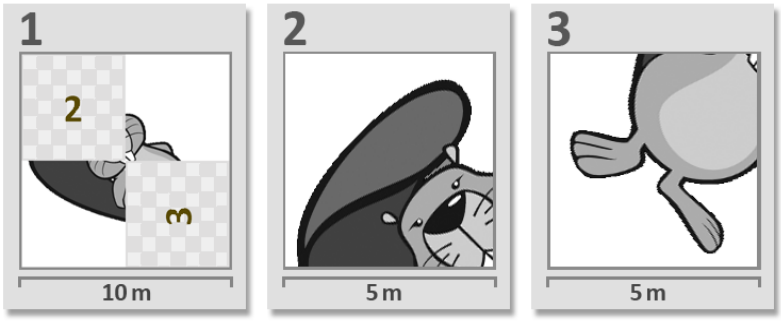
1. i 2. razred SŠ (Dabar)
3. i 4. razred SŠ (Stariji dabar)

Dabar i njegovi prijatelji su se dobrovoljno javili da pomognu obnovu Gradskog Muzeja Informatike. Dobili su zadatak da nacrtaju sliku na podu dimenzija 16 * 16 metara u jednoj od izložbenih prostorija.

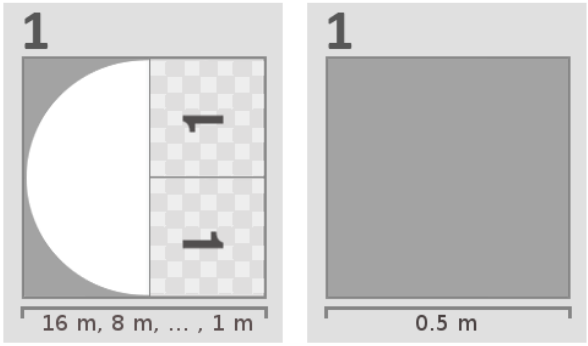
U Gradskom Muzeju informatike crtanje slika se vrši na osnovu određenih instrukcija.
Instrukcije se štampaju na određenim listovima papira oblika kvadrata. Svali list papira obeležen je brojem.

Na pojedinim listovima može se naći i prazno mesto na crtežu gde bi se ubacili crteži sa drugih listova. Na tim praznim mestima je prikazan broj i taj broj znači da se na tom mestu treba postaviti crtež sa lista papira na kome je taj broj. Svaki list ima dimenzije na dnu.

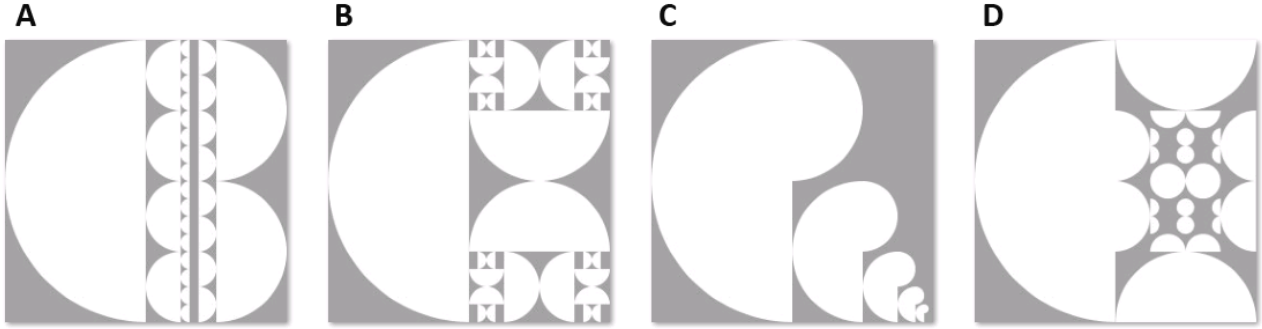
Evo primera za jedan crtež koji ima dimanzije 10*10 metara. Na crtežu se nalazi dabar.



Dabrovi su za crtanje poda dimenzija 16*16 metara dobili ovakve instrukcije:



Pitanje: Kako će izgledati nacrtan pod?



Obrazloženje

Tačan odgovor je pod B

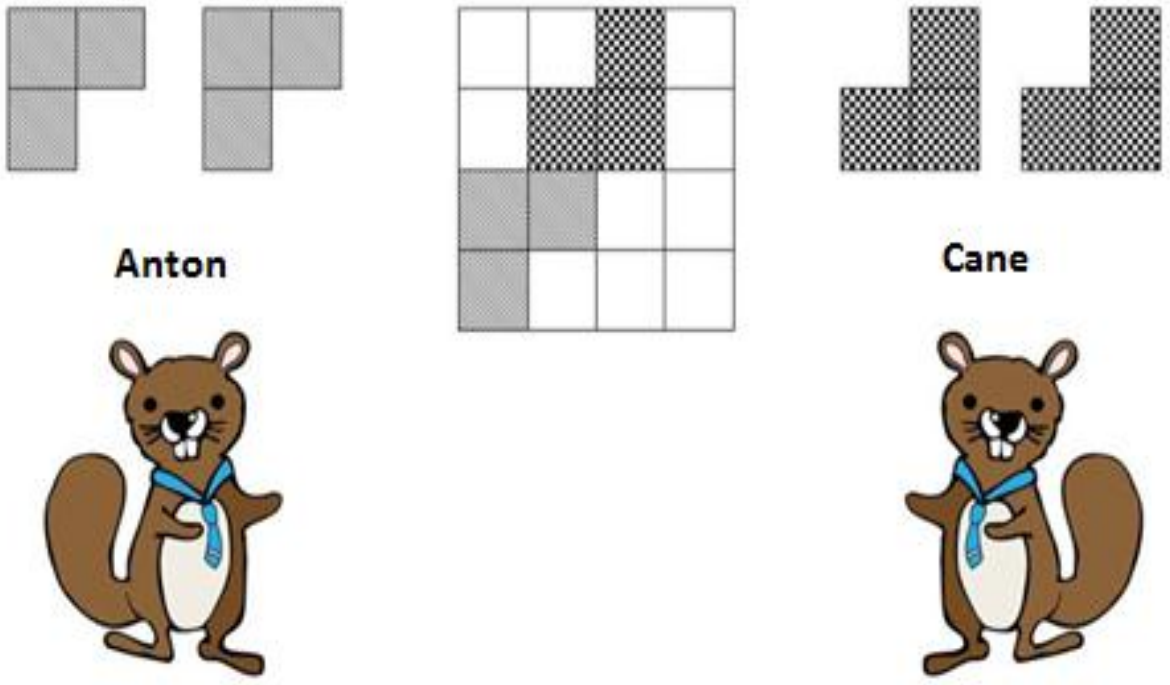
Informatička pozadina: Rekurzija je važan pojam u informatici. Rekurzija je funkcija koja poziva sama sebe. Rekurzivno programiranje je često korištena tehnikama kojom je moguće implementirati razne algoritme. Rekurzija uvek mora imati neko ograničenje koje će ju zaustaviti, jer bi se u suprotnom funkcija pozivala beskonačan broj puta, i program se nikada ne bi izvršio.

Anton i Cane igraju L-igru na 4x4 tabli. Oni se igraju tako što postavljaju L komadiće (komadići u obliku slova L), pri čemu prvo igra Anton a zatim Cane pa Anton…i to prema sledećim pravilima:

- svaki L komadić koji postavlja Anton je orijentisan kao što je prikazano na slici ispod,
- svaki L komadić koji postavlja Cane je orijentisan kao što je prikazano na slici ispod,
- svaki L komadić koji se postavlja na tablu ne sme da viri van table,
- dva L komadića se ne smeju preklapati.

L komadići se ne mogu pomerati nakon što se postave. Igrač gubi partiju kada je njihov red, ali ne može da stavi L komadić u skladu sa pravilima.

Primer gde Anton igra prvi je prikazan ispod. U ovom primeru, Anton može da dobije igru postavljanjem L komadića u donji desni ugao.



Pitanje: Anton može započeti igru na devet različitih načina. Koliko načina mu omogućava sigurnu pobedu, bez obzira kako se postavljaju naredni L komadići?

A) 0
B) 1
C) 2
D) 3

Obrazloženje

Tačan odgovor: B

Informatička pozadina

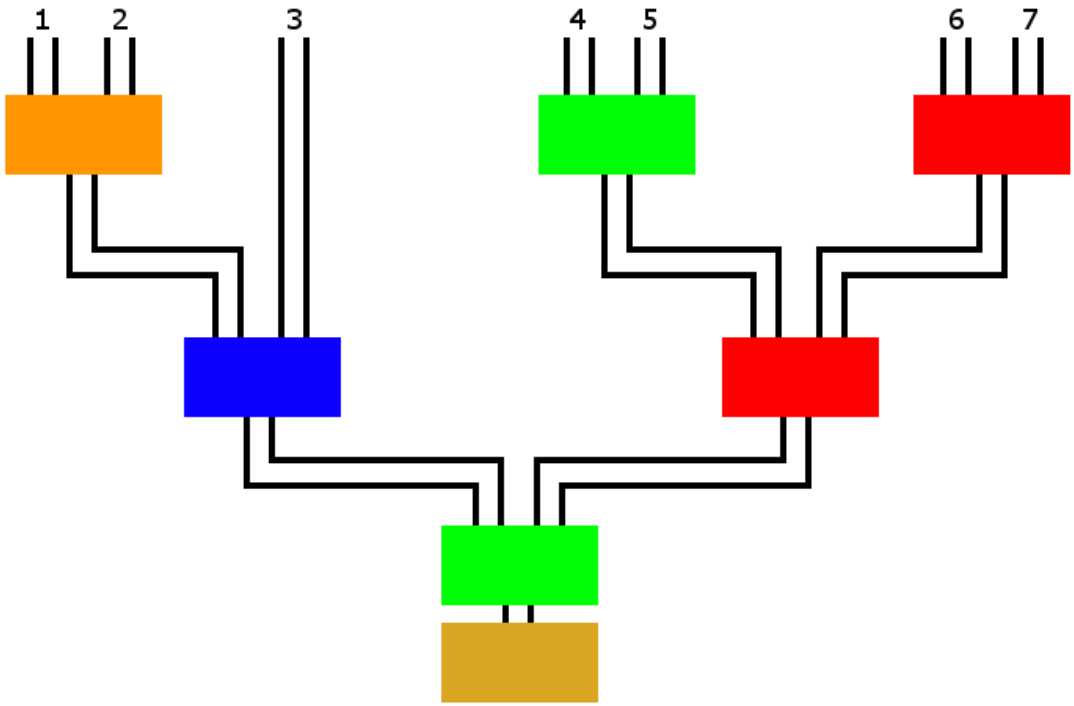
Sve mogućnosti u igri mogu se predstaviti pomoću dijagrama nalik onom u objašnjenju/rešenju. U ovom *stablu igre*, koreni čvor odgovara početnom stanju na tabli za igru. Onda za svaki mogući potez strelica ukazuje na dobijeno novo stanje na tabli. Potpuno stablo konstruiše se ako se nastavi tako na isti način za sve moguće poteze. Stablo igre je posebna vrsta *usmerenog grafa*.

Stablo igre može se konstruisati i pretraživati da bi se igrala ili analizirala igra. U zavisnosti od vrste problema, nekada je korisnije prvo istraživati susedne čvorove pre prelaska na susede na narednom nivou stabla (*pretraga u širinu – breadth-first search (BFS)*), dok je nekada korisnije istražiti što je dalje moguće (dublje po nivoima stabla) po svakoj grani pre „obrnute pretrage“ (backtracking-a), odnosno unazad do izbora najboljeg od svih mogućih poteza (*pretraga u dubinu – depth-first search (DFS)*). Ove dve strategije za pretragu imaju različita svojstva i zahteve za memoriju.

Dabar Žofri voli da se igra sa klikerima. On ima samo klikere crne i bele boje. Ali Žofri ima i mašinu koje se može proizvesti specijalni kliker zlatne boje.

Da bi dobio jedan kliker zlatne boje, Žofri mora da ubaci crne i bele klikere u sedam cevi koje vode do kutija određenih boja. Kutije su obeležene plavom, crvenom, narandžastom i zelenom bojom. Inače, kutije uvek primaju dva klikera a propuštaju jedan ili nijedan kliker.

Zlatni kliker će se dobiti samo ako u zlatnu kutiju uđe crni kliker.



Postoje četiri različite kutije (četiri različite boje) koje rade sledeće:

- **Plava:** Ako su oba klikera crna, ona propušta crni kliker. Ako su oba bela, nijedan ne prolazi. Ako su različitih boja, onda prolazi beli.
- **Crvena:** Ova kutija dozvoljava samo crnom klikeru da prođe ako su oba crni. Sve ostale kombinacije znače prolazak belog..
- **Zelena:** Ova kutija propušta samo beli kliker ako su oba bela. Sve ostale kombinacije kombinacije znače prolazak crnog.
- **Narandžasta:** Ova kutija propušta crni kliker ako su oba crna. Sve ostale kombinacije znače prolazak belog.

Pitanje: Koji od sledećih rešenja ne daju zlatni klker? :

a) 1. Beli 2. Beli 3. Crni 4. Crni 5. Beli 6. Crni 7. Crni

b) 1. Beli 2. Beli 3. Crni 4. Crni 5. Beli 6. Beli 7. Crni

c) 1. Beli 2. Beli 3. Crni 4. Beli 5. Beli 6. Crni 7. Crni

d) 1. Crni 2. Beli 3. Crni 4. Crni 5. Beli 6. Crni 7. Crni

Obrazloženje

Tačan odgovor: c) 1. Beli 2. Beli 3. Crni 4. Beli 5. Beli 6. Crni 7. Crni

Dva skenera kodiraju sliku prevođenjem njenih piksela u posebne šifre (kodiranje). Šifra se sastoji od broja svih uzastopnih piksela iste boje (crna / bela), zatim po broju svih uzastopnih piksela u drugoj boji, i tako dalje, počev od gornjeg levog ugla, sa leve na desnu stranu, red po red na niže.

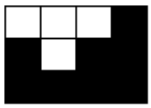
Ova dva skenera koriste različite metode da obrade kraj reda i početak sledećeg:

- **Skener A** obrađuje piksele red po red. Kada završi kodiranje u nekom redu, skener ponovo započinje kodiranje u sledećem redu.
- **Skener B** obrađuje piksele red po red. Kada završi kodiranje u nekom redu, skener samo nastavlja kodiranje u sledećem redu (ne počinje kodiranje iz početka)

Na primer, slika ispod će biti predstavljena sledećim kodovima:

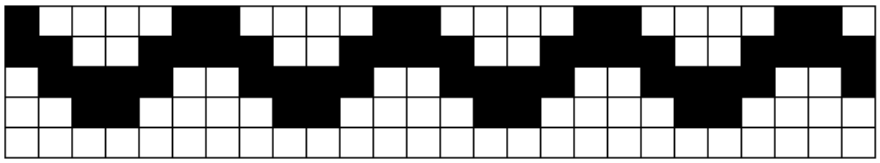
Skener A: 3,1,1,1,2,4 (3 belo, 1 crno, 1 crno, 1 belo, 2 crno, 4 crno)

Skener B: 3,2,1,6. (3 belo, 2 crno, 1 belo, 6 crno)

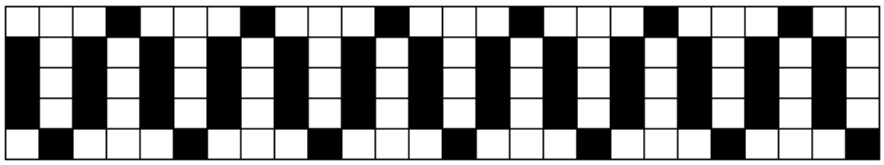


Pitanje: Koja od sledečih slika će imati isti kod bez obzira koji skener se koristi?

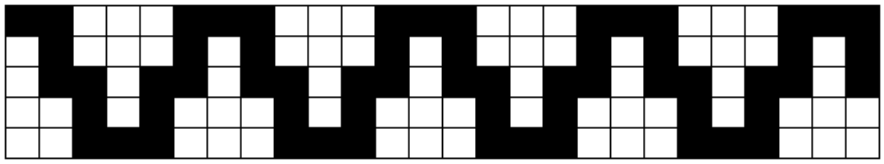
A:



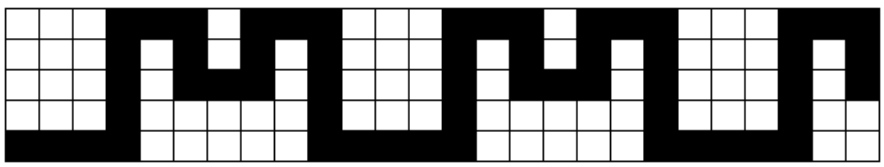
B.



C.



D.



Obrazloženje

Tačan odgovor je pod D

Skener je uređaj koji optički čita (ili skenira) sliku i pretvara je u digitalnu slikom. Kada se skenira, boja i osvetljenost svake male površine (piksela) koju senzor registruje, meri se i beleži kao numerička vrednost. Ovaj proces se naziva digitalizacija slike. Piksel je informatička reč nastala od pojma element slike (PICture ELement), jer je piksel najmanji element digitalne slike. Svaki piksel je uzorak originalne slike, a više uzoraka obezbeđuje tačniju- precizniju predstavu originalne slike.
Skener A koristi isprekidane linije da ponovi svoj kodiranje na narednim redovima dok skener B čita piksele kao jednu dugu neprekidnu sliku. Svaki može imati niz prednosti kada se koristi u praksi. Na primer, sa dugom slikom, možete koristiti manje brojeve sa skenerom B, ali biste takođe morali da kodirate dimenzije slike. To ne bi bilo praktično sa manjim slikama. Ovi kompromisi su veoma važne odluke koje se moraju doneti kada se radi u kompjuterskoj nauci.

Dobra drva

3. i 4. razred SŠ (Stariji dabar)

Tri dabra rade na uklanjanju istrulelih delova sa drvenih debla. Svaki dabar ima poseban zadatak:

- Toma meri rastojanje sa leve strane debla do prve tačke gde se drvo menja od istrulelog u dobro ili obrnuto, gde se drvo menja od dobrog u istrulelo.
- Za dobijeno rastojanje od Tome, Đole uklanja tu dužinu sa leve strane debla.
- Za dobijeno rastojanje, Karmen uklanja tu dužinu sa desne strane debla.

Danas oni rade na drvenom deblu dužine N metara. Drvo je istrulelo potpuno sa leve strane do neke tačke i potpuno istrulelo sa desne strane do neke tačke. Sve drvo između je u dobrom stanju.

Prvo Toma uradi svoj posao i izmeri X metara. Onda Toma dojavi Đoletu rastojanje X i Đole uradi svoj posao. Onda Toma ponovo uradi svoj posao na preostalom delu debla i izmeri Y.

Pitanje:

Koju vrednost bi trebalo Toma da dâ Karmen da bi završili započeti posao uklanjanja istrulelih delova drvenog debla?

A) X metara

B) Y metara

C) N-X metara

D) N-X-Y metara

Obrazloženje

Tačan odgovor je pod D) N-X-Y metara.

Posle prvog koraka, preostala dužina debla je dugačka N-X metara. Y metara sa leve strane debla je u dobrom stanju, tako da N-X-Y metara je istrulelo sa desne strane i ta vrednost treba da se prosledi Karmen da tu dužinu ukloni sa desne strane debla.

Informatička pozadina:

Dobijanje savršenog drveta liči na izdvajanje nekog dela teksta iz niske znakova. Npr., adrese e-pppošte su često u obliku idkorisnika@domen.org. Možda ćemo želeti da izdvojimo domen iz ovakve adrese e-pošte. Možemo posmatrati deo "idkorisnika@" i ".org" kao istrulele delove, a domen kao dobar deo drvenog debla. Da bi se ovo postiglo u programu za tabelarne proračune, možda ćemo moći iskoristiti ugrađene funkcije, koje se popnekad zovu FIND (*pronađi*), LEFT (*levo*) i RIGHT (*desno*), što je slično sa Tomom, Đoletom i Karmen tim redom. Programski jezici takođe omogućavaju da se urade ove operacije. Npr., u programskom jeziku Python postoji operator *slice* koji se koristi da se dobije deo teksta koji ostane kada se odbace levi i/ili desni deo niske znakova teksta.

Ključne reči:

funkcija u programu za tabelarne proračune, operacije sa niskom znakova teksta, operator *slice* u programskom jeziku Python.