

Programiranje 2

- odgovori na pitanja iz skripte -

Mina Milošević
mi17081@alas.matf.bg.ac.rs

2017/2018

1.1 Navesti prototip i opisati ponasanje funkcija malloc, calloc, realloc i free. U kom zaglavlju su deklarirane?

void *malloc(size_t n); - alocira blok memorije velicina n bajtova

void *calloc(size_t n, size_t size); - vraca pokazivac na memoriju velicine n objekata velicine size

void *realloc(void *memblock, size_t size); - promena velicine vec alociranog bloka memorije

void *free(void *p); - oslobadjanje memorije koja je alocirana pozivom predhodnih funkcija

Deklarisane se u <stdlib.h>

1.2 U kom segmentu memorije se tokom izvršavanja programa čuvaju dinamički alocirani blokovi memorije.

U hip segmentu

1.3 Kako se zove pojava kada se izgubi vrednost poslednjeg pokazivača na neki dinamički alocirani blok memorije? Zašto je ova pojava opasna?

Curenje memorije. Dolazi do prekida rada.

1.4 Sta je, nakon poziva free(p), vrednost pokazivaca p, a sta vrednost *p?

p ima istu vrednost kao na pocetku, a vrednost *p=0

1.5 Koja naredba treba da se, prema dobroj praksi, izvrši nakon free(p);?

p=NULL;

1.6 Opisati ukratko bar četiri česte greške koje se javljaju u vezi sa dinamičkom alokacijom memorije.

Curenje memorije – gubi se informacija o lokaciji dinamički alociranog a neoslobodjenog bloka memorije

Pristup oslobodjenoj memoriji – moguće je da naredni poziv (nakon free(p)) funkcije malloc vrati blok memorije na predhodnu poziciju

Oslobadjanje istog bloka vise puta - svaki naredni poziv free(p) za istu vrednost pokazivaca p prouzrokuje nedefinisano ponasanje programa

Oslobadjanje neispravnog pokazivaca – prosledjivanje pokazivaca na lokaciju koja pripada alociranom bloku a nije njegov pocetak

Prekoracenja i potkoracenja bafera – pristup elementima van granice bloka koji je dobijen

1.7 Kako se zove situacija u kojoj je memorija na hipu podeljena na mnoštvo malih i nepovezanih blokova?

Fragmentisanje memorije

2.1 Navesti faze razvoja softvera i ko ih obicno sprovodi.

- Planiranje – analiza (analiticar), modelovanje (projektant), dizajn resenja (programer)
- Realizacija – implementiranje, evaluacija, izrada dokumentacije
- Eksploatacija – obuka i tehnicka podrška, pustanje u rad, održavanje

2.2 Koje dve vrste dokumentacije treba da postoje?

Korisnicke i tehnicke

2.3 Opisati metodologije razvoja softvera.

- Metodologija vodopada - na sledecu fazu u razvoju softvera prelazi se tek kada je jedna potpuno završena
 - Metodologija iterativnog i inkrementalnog razvoja - razvoj se sprovodi u iteracijama i projekat se gradi inkrementalno. Iteracije obicno donose vise detalja i funkcionalnosti, a inkrementalnost podrazumeva dodavanje jednog po jednog modula. U jednom trenutku, vise razlicitih faza zivotnog ciklusa softvera moze biti u toku. U ovoj metodologiji vracanje unazad je moguće.
 - Metodologija rapidnog razvoja - faza planiranja svedena je na minimum zarad brzog dobijanja prototipova u iteracijama. Faza planiranja preklapa se sa fazom implementacije sto olaksava izmene zahteva u hodu. Proces razvoja kreće sa razvojem preliminarog modela podataka i algoritama.
 - Spiralna metodologija - kombinuje analizu rizika sa drugim metodologijama kao sto su metodologija vodopada i metodologije iterativnog razvoja. Spirala koja ilustruje ovu metodologiju prolazi vise puta kroz faze kao sto su planiranje, implementacija i evaluacija tekuceg verzije, kao i analiza rizika. Razlicite faze ne sprovode se istovremeno, vec jedna za drugom.
 - Agilna metodologija razvoja - tezi minimizovanju rizika razvijanjem softvera od strane visoko motivisanog tima, u iteracijama sa minimalnim dodavanjem funkcionalnosti i u kratkim vremenskim intervalima. Zahtevaju permanentnu komunikaciju, pozeljno uzivo.
 - *Skram* - vid agilne metodologije u kojem se neposredna, prakticna iskustva koriste u upravljanju izazovima i rizicima. Softverski proizvod sve vreme se održava u (integrisanom i testiranom) stanju koje se potencijalno moze isporučiti.
 - *Ekstremno progmiranje* - vid agilne metodologije u kojem su posebno vazne jednostavnost, motivacija i kvalitetni odnosi unutar tima. Programeri rade u parovima ili u vecim grupama. Sistem je integrisan i radi sve vreme
-

3.1 Navedite barem jedan alat za kontrolu verzija.

CVS, SVN, Git, Bazaar, Mercurial, Visual SourceSafe

3.2 Navedite dva nacina za imenovanje funkcije koja bi trebalo da se zove "izracunavanje finalne ocene".

int izracunavanje_finalne_ocene(); ili int izracunavanjeFinalneOcene();

3.3 Koje ime bi, u kamiljoj notaciji, nosila promenljiva int broj_cvorova?

int brCvorova

3.4 Koje ime bi, u madjarskoj notaciji, nosile promenljive float* X i int* broj_cvorova?

float *pfX; int *piBrojCvorova

3.5 Zaokruziti deklaracije koje su u skladu sa madjarskom notacijom:

char cKontrolniKarakter; char cKontrolni_karakter; char kontrolni_karakter; **int iKoeficijent;**
int iKamatnaStopa; int kamatna_stopa;

3.6 Navedite dva nacina za imenovanje promenljive koja oznacava ukupan broj clijenata.

int broj_klijenata; ili int brojKlijenata;

3.7 Koliko se preporucuje da najvise ima karaktera u jednoj liniji programa i koji su razlozi za to?

80, zbog preglednosti

3.8 Ukoliko neka linija naseg programa ima 300 karaktera, sta nam to sugerise?

Ta treba optimizovati kod tj. razbiti ga na funkcije.

3.9 Kada je prihvatljivo da jedna linija programa ima vise naredbi?

Ako su naredbe srodne.

3.10 U kojim situacijama se obicno neka linija programa ostavlja praznom?

Kako bismo razdvojili celine (npr. petlje).

3.11 Koliko belina zamenjuje jedan tab karakter?

Uglavnom 8

3.12 Zasto za nazublivanje teksta programa nije preporucljivo koristiti tabulator?

Zbog neuniformisanosti editora.

3.13 Sta su magicne konstante i da li one popravljaju ili kvare kvalitet programa?

Magicne konstante su svi brojevi sem 0 i 1. One kvare kvalitet programa

3.14 Kako se izbegava koriscenje „magicnih konstanti“ u programu?

Koriscenjem const char ili enum-a.

3.15 Sta je, kako bi kod bio lakši za održavanje, bolje koristiti umesto deklaracije `char ImeKorisnika[50]`?

```
const unsigned int MaksImeKorisnika = 50;  
char ImeKorisnika[MaksImeKorisnika];
```

3.16 Navedite barem jedan alat za automatsko generisanje tehnicke dokumentacije.

doxygen

3.17 Kojim se komentarom/markerom obično označava mesto u kodu na kojem treba dodati kod za neki podzadatak?

TODO

3.18 Koji se marker u okviru komentara u kodu obično koristi za označavanje potencijalnih propusta i/ili gresaka koje naknadno treba ispraviti?

FIXME

3.19 Koji je preporuceni obim jedne datoteke programa?

Do 200-300 linija maksimalno po datoteci.

4.1 Za sta se koristi matematička indukcija, a za sta rekurzija?

Indukcija je metod dokazivanja koji se koristi da se utvrdi tačnost izraza za sve prirodne brojeve.

Rekurzija se koristi za izračunavanje izraza.

4.2 Sta je potrebno da bi rekurzivna veza bila ispravno definisana?

Uslov za izlazak iz rekurzije (bazni slučaj) i rekurzivni korak.

4.3 Za svaki rekurzivni poziv, na programskom steku se stvara novi stek okvir za tu funkciju. (tacno/netacno)

4.4 Za svaki rekurzivni poziv, u kod segmentu se stvara nova kopija koda te funkcije. (tacno/netacno)

4.5 Za svaki rekurzivni poziv, na hipu se rezervise novi prostor za tu funkciju. (tacno/netacno)

4.6 Za svaki rekurzivni poziv, u segmentu podataka se rezervise novi prostor za njene promenljive. (tacno/netacno) - na stek segmentu se rezervise!

4.7 Navesti rekurzivnu verziju Euklidovog algoritma za računanje NZD:

```
unsigned nzd(unsigned a, unsigned b) {  
    if(b == 0)  
        return a;  
    else  
        return nzd(b, a%b);  
}
```

4.8 Dopuniti narednu funkciju koja iterativno racuna elemente Fibonacijevog niza:

```
int fib(int n) {  
    int f, fpp, fp;  
    if (n == 0 || n == 1) return n;  
    fpp = 0; fp = 1;  
    for(i = 2; i <= n; i++) {  
        ..... (int f=fpp+fp; fpp=fp; fp=f;)  
    }  
    return fp;  
}
```

4.9 Sta je repna rekurzija?

Repna rekurzija je rekurzija nakon koje nema dodatnih naredbi. Rekurzivan poziv je repno rekurzivni ako je vrednost rekurzivnog poziva upravo i konacan rezultat funkcije.

4.10 Kako se zove rekurzivni poziv nakon kojeg nema daljih akcija?

Repna rekurzija

4.11 Jedan od obrazaca funkcija kod kojih se moze eliminisati rekurzija je:

(a) glavna rekurzija; (b) repna rekurzija; (c) master rekurzija; (d) linearna rekurzija

4.12 Da li za bilo koju rekurzivnu funkciju postoji funkcija koja je njoj ekvivalentna a ne koristi rekurziju? Da li se rekurzija moze eliminisati iz bilo koje repno rekurzivne funkcije?

Da. Da

4.13 Sta su dobro a sta lose strane rekurzije?

Dobre strane: citljiv i kratak kod, jednostavan za analizu i razumevanje.

Lose strane: cena poziva (prilikom svakog poziva kreira se novi stek okvir i kopiraju se argumenti funkcije), suvisna izracunavanja (prilikom rastavljanja problema na manje potprobleme dolazi do ponovnog izracunavanja istih potproblema (fibonacijev niz), kako bi se ovo izbeglo koristimo tehniku memoizacije)

4.14 Izabrati jedan od dva ponudjena odgovora u zagradi (iterativno ili rekurzivno):

- 1) Ukoliko se razmatra brzina izvršavanja obično su povoljnije (iterativne /rekurzivne) implementacije algoritma.
- 2) Ukoliko se razmatra razumljivost obično je povoljnije (iterativne/rekurzivne) implementacije algoritma
- 3) Ukoliko se razmatra zauzeće memorije obično su povoljnije (iterativne/rekurzivne) implementacije algoritma

4.15 Napisati rekurzivnu funkciju za izracunavanje faktoriijela prirodnog broja:

```
int f(int x) {  
    if(x==1 || x==0)  
        return 1;  
    return x*f(x-1);  
}
```

4.16 Napisati rekurzivnu funkciju koja resava problem “kule Hanoja”.

```
void kule(unsigned n, char polazna, char dolazna, char pomocna) {  
    if (n > 0) {  
        kule(n-1, polazna, pomocna, dolazna);  
        printf("Prebaci disk sa kule %c na kulu %c\n", polazna, dolazna);  
        kule(n-1, pomocna, dolazna, polazna);  
    }  
}
```

5.1 Svaka instrukcija na racunaru se izvsava za $1 \cdot 10^{-9}$ s. Algoritam A1 za velicinu ulaza n zahteva n^2 instrukcija, a A2 zahteva n^3 instrukcija. Za koje velicine n ulaza algoritam A1, a za koje A2 moze da izvrsi rad za 1 minut?

10^{-9} s - vreme potrebno za izvsavanje jedne instrukcije. $A1=O(n^2)$, $A2=O(n^3)$

$$(n^2) \cdot 10^{-9} = 60$$

$$n^2 = 6 \cdot 1010$$

$$n = 100\,000 \cdot \sqrt{6}$$

$$(n^3) \cdot 10^{-9} = 60$$

$$n^3 = 6 \cdot 1010$$

$$n \sim 2154.5 \cdot \sqrt{6}$$

5.2 Svaka instrukcija na racunaru se izvsava za $2ns$. Algoritam A1 za obradu ulaza velicine n zahteva n^2 instrukcija, a A2 zahteva n^3 instrukcija. Sa obradom ulaza koje velicine algoritam A1/A2 moze da završi za jedan minut?

$2ns = 2 \cdot 10^{-9}$ s - vreme potrebno za izvsavanje jedne instrukcije. $A1=O(n^2)$, $A2=O(n^3)$

$$(n^2) \cdot 2 \cdot 10^{-9} = 60$$

$$n^2 = 3 \cdot 1010$$

$$n = 100\,000 \cdot \sqrt{3}$$

$$(n^3) \cdot 2 \cdot 10^{-9} = 60$$

$$n^3 = 12 \cdot 1010$$

$$n \sim 2154.5 \cdot \sqrt{12}$$

5.3 Kada se kaže da vazi $f(n) \in O(g(n))$?

Kada postoje c i N tako da: $f(n) \leq c \cdot g(n)$, za svako $n > N$.

5.4 Dokazati da vazi:

a) $n^2 + 2n = O(n^2)$

$n^2 + 2n \leq c \cdot n^2$, pretpostavljamo da vazi za neko $c \geq n_0$, uzimamo za n_0 da je 3

$$n^2 + 2n \leq 3 \cdot n^2, 2n \leq 2n^2, n \leq n^2, n^2 + 2n = O(n^2);$$

b) $2^n + n = O(2^n)$

$2^n + n \leq c \cdot 2^n$, $c \geq n_0$, uzimamo za n_0 da je 2

$$2^n + n \leq 2 \cdot 2^n, n \leq 2^n, 2^n + n = O(2^n)$$

c) $2^n + n^3 = O(2^n)$

$2^n + n^3 \leq c \cdot 2^n$, $c \geq n_0$, uzimamo za n_0 da je 2

$$2^n + n^3 \leq 2 \cdot 2^n, n^3 \leq 2^n, 2^n + n^3 = O(2^n)$$

d) $3n + 2n^2 = O(n^2)$

$3n + 2n^2 \leq c \cdot n^2$, $c \geq n_0$, uzimamo za n_0 da je 5

$$3n + 2n^2 \leq 5 \cdot n^2, 3n \leq 3n^2, n \leq n^2, 3n + 2n^2 = O(n^2)$$

e) $5^n + 2^n = O(5^n)$

$5^n + 2^n \leq c \cdot 5^n$, $c \geq n_0$, uzimamo za n_0 da je 2

$$5^n + 2^n \leq 2 \cdot 5^n, 2^n \leq 5^n, 5^n + 2^n = O(5^n)$$

5.5 Da li funkcija $7n^2 + 3$ pripada klasi:

a) $O(3n^2 + 1)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3 \log(n))$ e) $O(2 \cdot \log(n))$

5.6 Da li funkcija $3n \cdot \log(n) + 5n$ pripada klasi:

a) $O(n)$ b) $O(n \cdot \log(n))$ c) $O(n^2 \cdot \log(n))$ d) $O(n \cdot \log^2(n))$ e) $O(\log(n))$ f) $O(n + \log(n))$ g) $O(15n)$

5.7 Da li funkcija $7n^2 + 3n \cdot \log(n)$ pripada klasi:

a) $O(3n^2 + 1)$ b) $O(n)$ c) $O(n^2)$ d) $O(n^3 \cdot \log(n))$ e) $O(2 \cdot \log(n))$

5.8 Da li funkcija $3n \cdot \log(n) + 5n + 100$ pripada klasi:

a) $O(n)$ b) $O(n \cdot \log(n))$ c) $O(n^2 \cdot \log(n))$ d) $O(n \cdot \log^2(n))$ e) $O(\log(n))$ f) $O(n + \log(n))$ g) $O(15n)$

5.9 Da li funkcija $n^6 + 2^n + 10^{10}$ pripada klasi:

a) $O(n)$ b) $O(2^n)$ c) $O(n^6)$ d) $O(n^{10})$ e) $O(10^{10})$ f) $O(n^6 + 2^n)$ g) $O(2^n + 10^{10})$ h) $O(2^n 10^{10})$

5.10 Ako $a(n)$ pripada klasi $O(n \log(n))$, a $b(n)$ pripada klasi $O(n^2)$, onda $a(n)+b(n)$ pripada klasi:

- a) $O(n \log(n))$ b) $O(n^2)$ c) $O(n \log(n)+n^2)$ d) $O(n^2 \log(n))$

5.11 Ako $a(n)$ pripada klasi $O(n^2)$, a $b(n)$ pripada klasi $O(n^3)$, onda $a(n)+b(n)$ pripada klasi:

- a) $O(n^2)$ b) $O(n^3)$ c) $O(n^5)$ d) $O(n^6)$

5.12 Kada kazemo da je slozenost algoritma A jednaka $O(f(n))$?

$A = O(f(n))$ kada je $A(n) \leq c \cdot f(n)$

5.13 Ako je slozenost algoritma A za ulaznu vrednost n jednaka $O(n^3)$, a slozenost algoritma B za ulaznu vrednost n jednaka $O(n^4)$, kolika je slozenost algoritma C koji se izvrsava tako sto se izvrsava prvo algoritam A, pa B?

- a) $O(n^3)$ b) $O(n^4)$ c) $O(n^7)$ d) $O(n^{12})$

5.14 Odrediti n-ti clan niza $T(n)=x \cdot T(n-1)$, $T(0)=y$.

$T(n)=x^n \cdot y$

5.15 Kojoj klasi pripada resenje rekurentne relacije $T(n)=2T(n/2)+n$?

$O(n \cdot \log n)$

5.16 Kolika je slozenost algoritma za pronalazenje minimuma niza? Kolika je slozenost algoritma za pronalazenje drugog po velicini elementa niza?

$O(n)$. $O(n)$

5.17 Odrediti slozenost izvrsavanja sledece funkcije:

```
void f(int n) {  
    if (n<1)  
        printf("*");  
    else {  
        f(n-1);  
        printf("----\n");  
        f(n-1);  
    }  
}
```

$T(n) = 2 * T(n - 1) + c$

5.18 Odrediti slozenost izvrsavanja sledece funkcije:

```
void f(int n) {  
    if (n<2)  
        printf("* ");  
    else {  
        f(n-2);  
        f(n-1);  
        f(n-2);  
    }  
}
```

$T(n) = T(n-1) + 2 * T(n-2) + c$

5. 19 Za koji problem kazemo da pripada klasi P? Za koji problem kazemo da pripada klasi NP? Za koji problem kazemo da je NP-kompletno?

Problem je u klasi P ako je njegovo vreme izvršavanja $O(P(n))$ gde je $P(n)$ polinom po n .

Problem je u klasi NP ako postoji algoritam polinomske složenosti koji za proizvoljan ulaz može da proveriti da li vodi rešenju problema.

Problem je NP-tezak ako ako je svaki NP problem polinomski svodljiv na x .

Problem je NP-kompletno ako pripada klasi NP i ako je NP-tezak.

5.20 Kako se dokazuje da neki problem pripada klasi NP? Kakvu bi posledicu imao dokaz da neki NP-kompletno problem pripada klasi P a kakvu dokaz da neki NP-kompletno problem pripada klasi P?

Za proizvoljnu valuaciju proveravamo da li možemo da ga rešimo u polinomijalnom vremenu.

Posledica bi bila da je NP-tezak.

$NP = P$ / $NP \neq P$

5.21 Ako je svaki NP problem svodljiv u polinomskom vremenu na problem A, kakav je problem A?

On je NP-tezak

5.22 Ako je svaki NP-kompletno problem svodljiv u polinomskom vremenu na problem A, šta onda važi za problem A?

On je NP-kompletno

5.23 Ako neki NP-tezak problem A nije NP-kompletno, šta onda to znači?

On se ne pripada klasi NP.

5.24 Ako se zna da je algoritam A NP-kompletno i da algoritam B pripada klasi NP, kako se može dokazati da je algoritam B NP-kompletno?

Tako što pokazemo da svi problemi iz NP mogu da se svedu na njega u polinomijalnom vremenu, tj. da algoritam A može da se svede na problem B u polinomijalnom vremenu (teorema za NP-kompletno problem).

5.25 Dokazano je da važi:

a) $P \subset NP$

b) $NP \subset P$

c) $P = NP$

d) $P \neq NP$

5.26 Navesti primer problema koji pripada klasi P; klasi NP i problem koji je NP-kompletno.

N! / SAT / SAT

5.27 SAT problem je:

a) $P = NP$

b) $P \neq NP$

c) **problem iskazne zadovoljivosti**

d) problem ispitivanja složenosti programa

5.28 Sta je SAT? Da li je SAT NP-tezak problem? Da li je SAT NP-kompletno problem?

Problem iskazne zadovoljivosti. Da. Da.

5.29 Da li SAT pripada klasi P? Da li SAT pripada klasi NP? Sta je posledica tvrdjenja $SAT \in P$, a sta tvrdjenja $SAT \notin P$?

Ne zna se. Da. $P = NP$. $P \neq NP$

6.1 Sta je cilj verifikacije programa?

Dokazivanje ispravnosti programa

6.2 Testiranjem programa se može:

- a) dokazati da je program korektan za sve ulaze
- b) opovrgnuti da je program korektan za sve ulaze**
- c) dokazati da se program zaustavlja za sve ulaze
- d) dokazati da se program ne zaustavlja za neke ulaze**

6.3 Kako se zove provera korektnosti u fazi izvršavanja programa?

Dinamička verifikacija

6.4 Koji je najčešći vid dinamičke verifikacije programa?

Testiranje

6.5 Koliko bi, metodom iscrpnog testiranja, bilo potrebno izvršiti testova programa za sabiranje dva neoznačena 32-bitna cela broja?

$$2^{32} * 2^{32} = 2^{64}$$

6.6 U čemu je razlika između parcijalne i totalne korektnosti programa?

Parcijalna korektnost - program se zaustavlja i daje korektan rezultat

Totalna korektnost - ako se program zaustavlja za sve ulaze i ako je rezultat parcijalno korektan

6.7 Da li uz pomoć debagera može da se:

- a) efikasnije kompilira program
- b) lakše otkrije greška u programu**
- c) izračuna složenost izvršavanja programa
- d) registruje curenje memorije u programu
- e) umanji efekat fragmentisanja memorije

6.8 Kako se dokazuje korektnost rekurzivnih funkcija?

Matematičkom indukcijom

6.9 Kako se zove formula koja uključuje vrednosti promenljivih koje se javljaju u nekoj petlji i koja važi pri svakom ispitivanju uslova petlje?

Invarijanta petlje

6.10 Kako se zovu relacije koje se koriste u dokazivanju ispravnosti programa koji zadrže petlju?

Invarijanta petlje

6.11 Kako se interpretira trojka $\{\phi\}P\{\psi\}$?

Horova trojka.

Ako izvršavanje niza naredbi P počinje sa vrednostima ulaznih promenljivih koje zadovoljavaju uslov ϕ i ako P završi rad u konačnom broju koraka, tada vrednosti programskih promenljivih zadovoljavaju uslov ψ .

6.12 Navesti pravila Horovog formalnog sistema:

Aksioma dodele – semantika naredbe dodele

Pravilo posledice – moguće je ojačati preduslov, kao i oslabiti postuslov svake trojke

Pravilo kompozicije – semantika sekvencijalnog izvršavanja dve naredbe

Pravilo grananja – semantika if-then-else naredbe

Pravilo petlje – semantika while naredbe

6.13 Da li je u Horovoj trojci $\{\phi\}P\{\psi\}$, ϕ program ili formula, da li je P program ili formula, da li je ψ program ili formula?

Formula / Program / Formula

6.14 Dopuniti sledeću Horovu trojku tako da ona bude zadovoljena:

- | | |
|--|----------------|
| 1. $\{___\}$ if $(a > b)$ then $z := a$; else $z := b$; $\{z < 9\}$ | $a < 9, b < 9$ |
| 2. $\{___\}$ if $(a < b)$ then $z := a$; else $z := b$; $\{z > 0\}$ | $a > 0, b > a$ |
| 3. $\{___\}$ if $(a < b)$ then $z := a$; else $z := b$; $\{z > 3\}$ | $a > 3, b > 3$ |
| 4. $\{___\}$ $a := b$; $c := a$; $\{c > 3\}$ | $b > 3$ |
| 5. $\{___\}$ $n := x$; $\{n > 3\}$ | $x > 3$ |
-

7.1 Da bi se primenilo linearno pretrazivanje niza, elementi niza:

- a) ...neophodno je da budu sortirani
- b) ...pozeljno je da budu sortirani
- c) ...ne smeju da budu sortirani
- d) **...nepotrebno je da budu sortirani**

7.2 Da bi linearna pretraga bila primenljiva treba da vazi:

- a) niz mora da bude uredjen
- b) niz mora da bude niz brojeva
- c) **nema preduslova**

7.3 Koja je, po analizi najgoreg slucaja, slozenost za linearno pretrazivanje?

$O(n)$

7.4 Koja je slozenost linearne pretrage niza od k celih brojeva cije su vrednosti izmedju m i n?

$O(k)$

7.5 Napisati funkciju za binarno pretrazivanje niza celih brojeva.

```
int binarno(int *niz, int x, int l, int d){
    int sr;
    while(l<=d){
        sr=l+(d-l)/2;
        if(niz[sr]==x)
            return sr;
        else if(niz[sr]<x)
            l=sr+1;
        else
            d=sr-1;
    }
    return -1;
}
```

7.6 Da li binarno pretrazivanje radi:

- a) ...najbrze...
- b) ...najsporije...
- c) **...ispravno...**

samo ako su elementi niza sortirani?

7.7 Da bi se primenilo binarno pretrazivanje niza, elementi niza:

- a) **...neophodno je da budu sortirani**
- b) ...pozeljno je da budu sortirani
- c) ...ne smeju da budu sortirani
- d) ...nepotrebno je da budu sortirani

7.8 Da bi binarna pretraga bila primenljiva treba da vazi:

- i) **niz mora da bude uredjen**
- ii) niz mora da bude niz brojeva
- iii) nema preduslova

7.9 Da li je binarno pretrazivanje moguće primeniti ako su elementi niza sortirani opadajuće?

Da

7.10 Koja je, po analizi najgoreg slucaja, slozenost algoritma za binarno pretrazivanje niza?

$O(\log(n))$

7.11 Koja je slozenost binarne pretrage niza od k celih brojeva cije vrednosti su izmedju m i n?

$O(\log(k))$

7.12 Koja komanda sledi iza komande *if(data[mid]==value)* u funkciji koja binarnom pretragom trazi vrednost value u nizu data?

`return mid`

7.13 Ako je niz od 1024 elementa sortira, onda binarno pretrazivanje moze da proveriti da li se neka vrednost nalazi u nizu u:

a) 10 koraka b) 128 koraka c) 512 koraka d) 1024 koraka

7.14 Nesortirani niz ima n elemenata. Potrebno je proveravati da li neka zadata vrednost postoji u nizu:

- a) $O(1)$ puta: da li se tada vise isplati primenjivati linearnu ili binarnu pretragu?
 - b) $O(n)$ puta: da li se tada vise isplati primenjivati linearnu ili binarnu pretragu?
 - c) $O(n^2)$ puta: da li se tada vise isplati primenjivati linearnu ili binarnu pretragu?
- Mora da se primeni linearna pretraga jer je niz nesortiran.

7.15 Navesti prototip funkcije `bsearch` iz `stdlib.h`.

```
void *bsearch(const void *key, const void *base, size_t n, size_t size,
              int (*compare)(const void *el1, const void *el2));
```

7.16 Dopuniti funkciju tako da moze da se koristi kao poslednji element funkcije `bsearch` za pretragu niza brojeva tipa `int`:

```
int poredi(const void* px, const void* py) {
    return *((int*)px) - *((int*)py);
}
```

7.17 Koji uslov je dovoljan da bi metodom polovljenja intervala mogla da se aproksimira nula funkcije f, ako funkcija f ima razlicit znak na krajevima intervala?

Uslov da je funkcija f neprekidna na tom intervalu.

7.18 Kako se jednostavno proverava da funkcija f ima razlicit znak na krajevima intervala [a, b]?

Proverimo znak od $f(a) * f(b)$.

7.19 Metod polovljenja intervala koji trazi nulu neprekidne funkcije f na zatvorenom intervalu [a,b], ima smisla primenjivati ako je funkcija f neprekidna i vazi :

a) $f(a) * f(b) < 0$ b) $f(a) * f(b) > 0$ c) $f(a) + f(b) > 0$ d) $f(a) + f(b) < 0$

7.20 Kada se zaustavlja numericko odredjivanje nule funkcije metodom polovljenja intervala?

Ako se nadje nula funkcije ili ako se upadne u datu epsilon okolinu nule funkcije.

7.21 Napisati prototip funkcije koja odredjuje nulu zadate funkcije na intervalu [a,b] (jedan od argumenata treba da bude pokazivac na funkciju).

f(double levi_kraj, double desni_kraj, double epsilon, double (*f)(double))

7.22 Koje su dve operacije osnovne u vecini algoritama sortiranja?

Operacija poredjenja i razmene dva elementa niza

7.23 Navesti imena bar pet algoritama sortiranja i njihovu slozenost u najgorem i prosečnom slučaju.

Bubble sort, insertion sort, selection sort – $O(n^2)$

Shell sort – $O(n^2)$, $O(n \log^2 n)$

Merge sort – $O(n \log n)$, $O(n)$

Quick sort – $O(n^2)$, $O(n \log n)$

7.24 Da li postoji algoritam za sortiranje čija se slozenost u prosečnom i najgorem slučaju razlikuju?

Da (quick i merge sort)

7.25 Opisati osnovnu ideju algoritma selection sort.

Osnovna ideja je da se jedan element postavi na svoje mesto, a zatim da se isti metod rekursivno primeni na niz koji je za jedan kraci od polaznog.

7.26 Prikazati stanje niza prilikom izvršavanja selection sort-a za niz 5 3 4 2 1.

(5 3 4 2 1) → (1 3 4 2 5) → (1 2 4 3 5) → (1 2 3 4 5) → (1 2 3 4 5)

7.27 Sta vazi nakon i-tog prolaska kroz petlju u algoritmu selection sort?

i-ti element po veličini se dovodi na poziciju i

7.28 Za niz dužine n, koliko najefikasnije implementacije selection sort algoritma vrše poredjenja i zamena?

Broj poredjenja je $O(n^2)$, a broj razmena je $n-1$ tj. $O(n)$

7.29 Koji je slučaj najgori za algoritam sortiranja selection sort?

$O(n^2)$

7.30 Opisati osnovnu ideju algoritma bubble sort.

U svakom prolasku kroz niz poredi uzastopne elemente i razmenjuje ih mesta ako su u pogresnom poretku. Prolasak kroz niz se ponavlja sve dok se ne napravi prolaz u kome nije bilo razmena.

7.31 Koji ulazni niz predstavlja najgori slučaj za algoritam bubble sort?

Najgori slučaj je kada su elementi u suprotnom poretku od trazenog - $O(n^2)$

7.32 Opisati osnovnu ideju algoritma insertion sort.

Niz se sortira tako sto se jedan po jedan element niza umeće na odgovarajuće mesto u do tada sortirani deo niza.

7.33 Prikazati stanje niza prilikom izvršavanja insertion sort-a za niz 5 3 4 1 2.
(5 3 4 1 2) → (3 5 4 1 2) → (3 4 5 1 2) → (1 3 4 5 2) → (1 2 3 4 5)

7.34 Kolika je složenost insertion sort-a u najgorem i u prosečnom slučaju?
 $O(n^2)$

7.35 Dopuniti implementaciju funkcije *umetni* koja se koristi u algoritmu za sortiranje insertion sort:

```
void umetni(int a[], int i){
    int j;
    for(_____ ) → (j=i; j>0 && a[j]<a[j-1]; j--)
        razmeni(a, j, j-1);
}
```

7.36 Opisati osnovnu ideju algoritma merge sort.

Deli niz na dva dela čija se dužina razlikuje najviše za 1, rekurzivno sortira svaku od njih, i zatim objedinjuje sortirane polovine. Za objedinjavanje je neophodno koristiti dodatni niz pomoćni niz.

7.37 Kod algoritma merge sort je:

- a) **jednostavno razdvajanje na dva dela niza koji se sortira, ali je komplikovano spajanje**
- b) komplikovano razdvajanje na dva dela niza koji se sortira, ali je jednostavno spajanje
- c) jednostavno je i razdvajanje na dva dela niza koji se sortira i njihovo spajanje
- d) komplikovano je i razdvajanje na dva dela niza koji se sortira i njihovo spajanje

7.38 Dopuniti implementaciju algoritma merge sort:

```
void mergesort_(int a[], int l, int d, int tmp[]){
    if (l < d){
        int i, j;
        int n = d - l + 1, s = l + n/2;
        int n1 = n/2, n2 = n - n/2;
        _____ → mergesort_(a, l, s-1, tmp);
        _____ → mergesort_(a, s, d, tmp);
        merge(a+l, n1, a+s, n2, tmp);
        for (i = l, j = 0; i <= d; i++, j++)
            a[i] = tmp[j];
    }
}
```

7.39 Opisati osnovnu ideju algoritma quick sort.

Modifikacija osnovne ideje selection sort-a tako što umesto min/max, u svakom koraku na svoje mesto dovodi neki element (pivot) koji je relativno blizu sredine niza. Potrebno je prilikom dovodjenja pivota na svoje mesto grupisati sve elemente manje od njega levo od njega, a sve elemente veće od njega desno od njega. Ključni korak quick sort je tzv. korak particionisanja koji nakon izbora nekog pivotirajućeg elementa podrazumeva da se niz organizuje da prvo sadrži elemente manje od pivota, zatim pivotirajući element, i na kraju elemente veće od pivota.

7.40 U okviru algoritma quick sort, kada se izabere pivot, potrebno je izvršiti:

- a) permutovanje elemenata niza
- b) particionisanje elemenata niza**
- c) invertovanje elemenata niza
- d) brisanje elemenata niza

7. 41 Kod algoritma quick sort je:

- a) jednostavno razdvajanje na dva dela niza koji se sortira, ali je komplikovano spajanje
- b) komplikovano razdvajanje na dva dela niza koji se sortira, ali je jednostavno spajanje**
- c) jednostavno je i razdvajanje na dva dela niza koji se sortira i njihovo spajanje
- d) komplikovano je i razdvajanje na dva dela niza koji se sortira i njihovo spajanje

7.42 Dopuniti implementaciju algoritma quick sort:

```
void qsort_(int a[], int l, int d){  
    if (l<d){  
        razmeni(a, l, izbor_pivota(a, l, d));  
        int p = partitionisanje(a, l, d);  
        _____ → qsort_(a, l, p-1);  
        _____ → qsort(a, p+1, d);  
    }  
}
```

7.43 Koja je slozenost koraka particionisanja (za niz od n elemenata) koje se koristi u algoritmu quicksort?

$O(n)$

7.44 Koji ulazni niz predstavlja najgori slucaj za algoritam quick sort?

Sortirani niz

7.45 Koja je slozenost u najgorem slucaju algoritma quick sort:

- a) ako se za pivot uzima prvi element niza? - $O(n^2)$
- b) ako se za pivot uzima poslednji element niza? - $O(n^2)$
- c) ako se za pivot uzima srednji (po indeksu) element niza? $O(n\log(n))$

7.46 Navesti bar dva algoritma sortiranja iz grupe “podeli i vladaj”.

Merge i quick sort

7.47 Kojem tipu algoritama (ne po klasi slozenosti, nego po pristupu) pripadaju algoritmi za sortiranje quick-sort i merge sort? Sta je tesko a sta lako u prvom, a sta u drugom?

“divide and conquer” . Quick sort - tesko je particionisanje, a lako spajanje; Merge sort – obrnuto.

7.48 Da li je u algoritmu quick sort razdvajanje na dva podniza lako ili tesko? Da li je u algoritmu quick sort spajanje sortirana dva podniza lako ili tesko?

Tesko. Lako

7.49 Navesti prototip funkcije qsort iz <stdlib.h>.

```
void qsort(void *base, size_t number, size_t size, int (*compare)(const void *el1, const void *el2));
```

7.50 Navesti primer funkcije compare koja se moze koristiti u okviru qsort:

```
int compare(const void* a, const void* b){  
    double x=((double*)a);  
    double y=((double*)b);  
    if(x<y) return 1;  
    else if(x>y) return -1;  
    return 0; }
```

7.51 Ako se funkcija qsort iz standardne biblioteke koristi za sortiranje niza struktura tipa S po članu *kljuc* tipa int, navesti funkciju za poredjenje koju treba proslediti funkciji qsort.

```
int poređi(const void* a, const void* b){
    S x=(S*)a;
    S y=(S*)b;
    if(x.kljuc > y.kljuc) return 1;
    else if(x.kljuc < y.kljuc) return -1;
    return -1;
}
```

7.52 Dopuniti implementaciju funkcije za racunanje vrednosti polinoma:

```
double vrednost_polinoma(double x, double a[], int n) {
    int i;
    double v=a[n];
    for (i=n-1; i>=0; i--)
        _____ → v = x*v + a[i];
    return v;
}
```

7.53 Koja je leksikografski sledeca varijacija sa ponavljanjem skupa {1, 2, 3} duzine 4 u odnosu na varijaciju 2313?

2321

7.54 Dopuniti naredni kod koji generise sledecu varijaciju duzine k od elemenata 1, 2,..., n:

```
int sledeca_varijacija(int a[], int n, int k) {
    int i;
    for (i=k-1; i>=0 && a[i]==____; i--) → a[i]=n;
    a[i] = _____; → a[i]=1;
    if (i < 0)
        return 0;
    a[i]++;
    return 1;
}
```

7.55 Dopuniti naredni kod tako da lista sve varijacije sa ponavljanjem duzina k brojeva izmedju 1 i n:

```
void varijacije_(int a[], int i, int n, int k) {
    int j;
    if(i==k)
        ispisi(a, k);
    else
        for(j=1; j<=n; j++) {
            _____ → a[i]=j;
            _____ → varijacije_(a, i+1, n, k);
        }
}
```

7.56 Data je permutacija 41532. Koja permutacija je sledeca u leksikografskom poretku?

42135

7.57 Dopuniti narednu funkciju tako da ona ispisuje sve kombinacije duzine k od elemenata 1, 2,..., n:

```
void kombinacije_(int a[], int i, int k, int min, int n) {
    if (i==k)
        ispisi(a, k);
    else
        if _____ → (k-i <= n-min+1){
            a[i] = min;
            kombinacije_(a, i+1, k, min+1, n);

            kombinacije_(a, i, k, min+1, n);
        }
}
```

7.58 Dopuniti naredni kod koji generise sledecu kombinaciju duzine k od elemenata 1, 2,..., n:

```
int sledeca_kombinacija(int a[], int k, int n) {
    int i=k-1;
    while(i>=0 && a[i]==n-k+i+1)
        _____; → i--;
    if (i<0)
        return 0;
    a[i]++;
    for (i=i+1; i<k; i++)
        a[i] = ____; → a[i]=a[i-1]+1;
    return 1;
}
```

7.59 Navesti sve particije broja 4.

4 | 3 1 | 1 3 | 2 2 | 2 1 1 | 1 2 1 | 1 1 2 | 1 1 1 1

7.60 Ako je promenljiva tipa unsigned int, kada izraz $n \& (n-1)$ ima vrednost 0?

Kada je n stepen dvojke .

7.61 Ukoliko broj n nije nula, cemu je jednaka vrednost izraza $n \& (n-1)$?

Izraz $n \& (n-1)$ ce na mesto prve jedinice, od mesta najmanje tezine, staviti 0

7.62 Koju vrednost ima izraz $0xA3 \& 0x24 \ll 2$?

$00100100 \ll 2 = 10010000 \rightarrow 10100011 \& 10010000 = 10000000 \rightarrow 0x80$

7.63 Koju vrednost ima izraz $0xB8 | 0x51 \ll 3$?

$01010001 \ll 3 = 10001000 \rightarrow 10111000 | 10001000 = 10111000 \rightarrow 0xB8$

7.64 Koju vrednost ima izraz $0x12 \wedge 0x34$?

$00010010 \wedge 00110100 = 00100110 \rightarrow 0x26$

7.65 Koju vrednost ima izraz $0x34 \ll 2$?

$00110100 \ll 2 = 11010000 \rightarrow 0xD0$

7.66 Koju vrednost ima izraz $(13 \& 17) \ll 2$?

$00001101 \& 00010001 = 00000001 \rightarrow 00000001 \ll 2 = 00000100 \rightarrow 4$

7.67 Napisati izraz cija je vrednost broj koji se dobije invertovanjem poslednjeg bita broja x?

$x \wedge 1$

7.68 Napisati izraz koji vraca treci bit najmanje tezine celog broja n.

$n \& (1 \ll 2) ? 1 : 0;$

7.69 Napisati izraz (ne funkciju!) koji je jednak vrednosti c tipa unsigned char u kojoj su promenjenje vrednosti tri poslednja bita.

$c \wedge (\sim(0 \ll 3));$

7.70 Napisati izraz cija je vrednost broj koji se dobija tako sto se upisu sve jedinice u poslednji bajt u zapisu neoznacene celog broja x.

$x | (\sim(0 \ll 8));$

7.71 Neka su x i y tipa unsigned char. Napisati izraz koji je tacan akko su razliciti poslednji bit vrednosti x i pocetni bit vrednosti y.

$(x \& (1 \ll \text{sizeof}(\text{unsigned char}) * 8 - 1)) != (y \& 1) \ll (\text{sizeof}(\text{unsigned char}) * 8 - 1)$

7.72 Neka su x i y tipa unsigned char. Napisati izraz koji je tacan akko su jednake vrednosti bitova na trecem bitu najmanje tezine za x i y.

$(x \& (1 \ll 2)) == (y \& (1 \ll 2))$

7.73 Neka su x i y tipa unsigned char. Napisati izraz koji je tacan akko su razlicite vrednosti bitova na osmom bitu najmanje tezine za x i y.

$(x \& (1 \ll 7)) != (y \& (1 \ll 7))$

7.74 Ako je c tipa unsigned char, napisati izraz koji vraca vrednost u kojoj je na poslednjem bitu odgovarajuci bit vrednosti c, a svi ostali bitovi su jednaki 0.

$c \& (1 \ll \text{sizeof}(\text{unsigned char}) * 8 - 1);$

7.75 Ako je c tipa unsigned char, napisati izraz koji vraca vrednost u kojoj je na preposlednjem bitu odgovarajuci bit vrednosti c, a svi ostali bitovi su jednaki 1.

$c | (\sim(1 \ll (\text{sizeof}(\text{unsigned char}) * 8 - 2)));$

7.76 Neka je x 32-bitan neoznaceni ceo broj, a b neoznaceni karakter. Izraz kojim se b inicijalizuje na vrednost bajta najvece tezine broja x je:

$b = x \gg 24;$

8.1 Koliko najmanje bajtova moze da zauzima jedan element povezane liste?

$\text{sizeof}(\text{pokazivac}) + 1$

8.2 Napisati rekurzivnu funkciju void obrisi(cvor *l); koja brise sve elemente jednostruko povezane liste.

```
void obrisi(cvor *l){
    if(pocetak){
        obrisi(l->sledeci);
        free(l);
    }
}
```

8.3 Ukoliko se znaju pokazivaci na prvi i poslednji element jednostruko povezane liste, koja je slozenost operacije brisanja prvog, a koja slozenost operacije brisanja poslednjeg elemanta?

$O(1)$. $O(n)$

8.4 Poznata je adresa pocetnog cvora jednostruko povezane liste. Koja je slozenost operacije uklanjanja njenog poslednjeg cvor, ako nije a koja ako jeste poznata njegova adresa?

$O(n)$. $O(n)$

8.5 Dopuniti implementaciju f-je koja dodaje element na pocetak povezane liste:

```
int dodaj_na_pocetak(cvor **pokazivac_na_pocetak, cvor *novi){
    novi->sledeci = *pokazivac_na_pocetak;
    _____ → *pokazivac_na_pocetak = novi;
    return 0;
}
```

8.6 Koja struktura podataka moze da modeluje lift?

Stek (LIFO)

8.7 Stek je?

LIFO FILO FIFO LILO

8.8 Koje su osnovne operacije steka i koja je njihova slozenost u implementaciji baziranoj na listama?

Osnovne operacije: *push* (dodaje elemente na vrh steka) i *pop* (skida elemente sa vrha steka). $O(1)$

8.9 Ako su s1 i s2 prazni stekovi, sta je njihova vrednost nakon operacija push(s1, 1), push(s2, 2), push(s1, 3), push(s2, pop(s1))?

S1 1. S2 2 i 3

8.10 Koje su slozenosti operacije dodavanja na dno steka i dodavanje na kraj reda?

$O(n)$, $O(1)$

8.11 Dopuniti implementaciju funkcije push koja dodaje element na vrh steka implementiranog koriscenjem povezane liste:

```
int push(cvor **pokazivac_na_vrh, int podatak){
    cvor* novi = novi_element(podatak);
    _____ → if(novi == NULL)
        return -1;
    return dodaj_na_pocetak(pokazivac_na_vrh, novi);
}
```

8.12 Dopuniti implementaciju funkcije pop koja skida element sa vrha steka implementiranog koriscenjem povezane liste:

```
int pop(cvor **pokazivac_na_vrh, int *podatak){
    _____ → return obrisi_sa_vrha(pokazivac_na_vrh, podatak);
}
```

8.13 Koje su osnovne operacije reda i koja je njihova slozenost u implementaciji baziranoj na listama?

Osnovne operacije: *add* (dodaje elemente na kraj reda) i *get* (skida elemente sa pocetka reda). $O(1)$

8.14 Navesti implementaciju funkcije get koja cita i brise element sa kraja reda.

```
int get(cvor** pocetak, cvor** kraj, int *broj) {
    cvor *tmp;
    if (*pocetak== NULL)
        return -1;
    tmp = *pocetak;
    *broj = tmp->broj;
    *pocetak= tmp->sledeci;
    _____ → if (*pocetak == NULL)
    _____ → *kraj = NULL;
    free(tmp);
    return 0;
}
```

8.15 Red je?

LIFO FILO **FIFO** **LIFO**

8.16 Sta karakterise red ako je implementiran koriscenjem povezanih lista?

Pokazivac na pocetak reda i pokazivac na kraj reda.

8.17 Koja je slozenost operacije brisanja poslednjeg elemanta jednostruko povezane a koja dvostruko povezane liste?

$O(n)$. $O(1)$

8.18 Navesti primer strukture koja opisuje elemente dvostruko povezane liste.

```
typedef struct cvor_liste {
    int broj;
    struct cvor_liste *prethodni;
    struct cvor_liste *sledeci;
} cvor;
```

8.19 Opisati strukturu kruzna lista.

U kruznoj listi, poslednji element liste ukazuje na prvi element liste. To omogućava da se do bilo kog elementa može doći posavši od bilo kog drugog elementa.

```
typedef struct cvor_liste {  
    int broj;  
    struct cvor_liste *sledeci;  
}cvor;
```

8.20 Ako binarno stablo ima dubinu 9, koliko najviše cvorova ono može da ima?

Zavisi da li smatramo da je cvor na dubini 0 ili 1. Recimo da smatramo da je cvor na dubini 1, a da je ukupna dubina n . U tom slučaju, to je $2^9 - 1$, a u slučaju da smatramo da je cvor na dubini 0, a ukupna dubina je n , to je $2^{10} - 1$.

8.21 Ako binarno stablo ima 15 cvorova, koja je njegova najmanja moguća a koja najveća moguća dubina?

a) 3 i 14 b) 3 i 16 c) 5 i 14 d) 5 i 16

8.22 Navesti definiciju strukture koja opisuje cvor binarnog stabla.

```
typedef struct cvor{  
    int podatak;  
    struct cvor *levo;  
    struct cvor *desno;  
}cvor;
```

8.23 Koje vrste obilaska u dubinu binarnog stabla postoje?

Infiksno, prefiksno i postfiksno

8.24 Sta je to uređeno binarno stablo?

U uređenom binarnom stablu za svaki cvor n , vrednosti svakog cvora iz njegovog levog podstabla su manje ili jednake od vrednosti u cvoru n , a vrednosti svakog cvora iz njegovog desnog podstabla su veće od vrednosti u cvoru n .

8.25 U uređenom binarnom stablu sve vrednosti u cvorovima levog podstabla cvora X su manje ili jednake od vrednosti _____.

cvora X

8.26 Koja je složenost dodavanja novog elementa u uređeno binarno stablo?

Koja je složenost pronalazenja elementa u uređenom binarnom stablu?

$O(n)$. $O(n)$

8.27 Koja je složenost dodavanja elementa u uređeno binarno stablo koje ima k elemenata, a n nivoa?

$O(n)$

8.28 Koja dodatna struktura podataka se koristi prilikom nerekurzivne implementacije obilaska stabla u sirinu?

Red

8.29 Da li se postfiksni ispisivanjem elemenata binarnog stabla dobija isti niz elemenata koji se dobija prefiksni obilaskom ali u obrnutom poretku?

Ne

8.30 Dato je stablo:

d	Prefiksni obilazak (K-L-D):	d, e, c, a, f, b
/ \	Infiksni obilazak (L-K-D):	c, a, e, d, f, b
e f	Postfiksni obilazak (L-D-K):	c, a, e, b, f, d
/\ \	Obilazak u sirinu:	d, e, f, c, a, b
c a b		

8.31 Koja je složenost operacije ispitivanja da li element pripada proizvoljnom uredjenom binarnom stablu? Koja je složenost operacije ispitivanja da li element pripada balansiranom uredjenom binarnom stablu? Ukoliko je dato binarno stablo sa n cvorova, koja je složenost operacije konstruisanja istog takvog stabla?

$O(n)$. $O(\log n)$. $O(n \log n)$

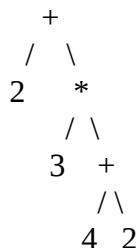
8.32 Složenost operacije pronalazenja elementa u uredjenom binarnom stablu zavisi od strukture stabla. Kada je ta složenost najmanja a kada najveća?

Ako je degenerisano, $O(n)$. Ako je balansirano, $O(\log n)$.

8.33 Kako se predstavljaju izrazi u vidu stabla?

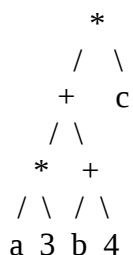
Matematički izrazi prirodno se mogu reprezentovati u vidu stabla. Na primer, izraz $3*(4+5)$ može se reprezentovati kao stablo u kojem je korenu $*$, sa levim podstablom 3, i sa desnim potomkom cvor $+$, koji ima potomke 4 i 5.

8.34 Binarnim stablom predstaviti izraz $2+3*(4+2)$.



8.35 Prikazati u vidu binarnog stabla izraz koji ima sledeći prefiksni zapis:

$* + * a 3 + b 4 c$.



8.36 Prikazati u vidu stabla izraz koji ima sledeci prefiksni zapis:

+ * * x 2 + x 3 7.

