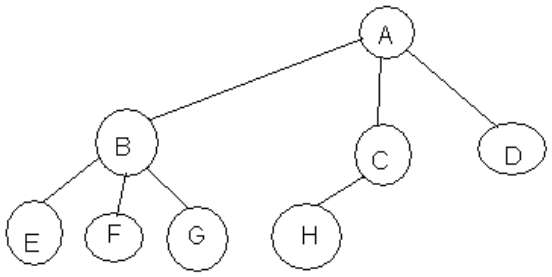
Algoritmi i strukture podataka
3. blic – pitanja s foruma
ak. god. 2005/06

---

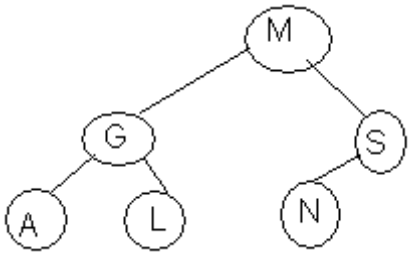
1. Stablo ima sljedeće atribute:



RJ: stupanj = 3, dubina = 3, potpuno

---

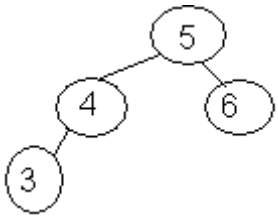
2. Zadana je funkcija (ne sjećam se koja), što se ispisuje?



RJ: G M

---

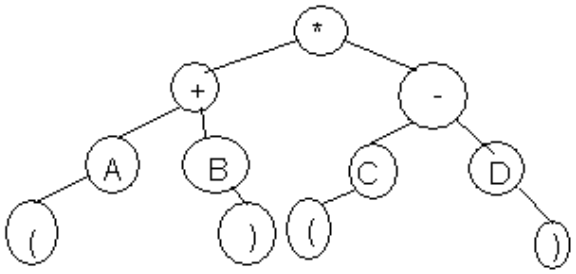
3. Zadano je 5, 4, 6, 3 i napravi se stablo i ima funkcija i treba znat što ispisuje:



RJ: 3 4 5 6

---

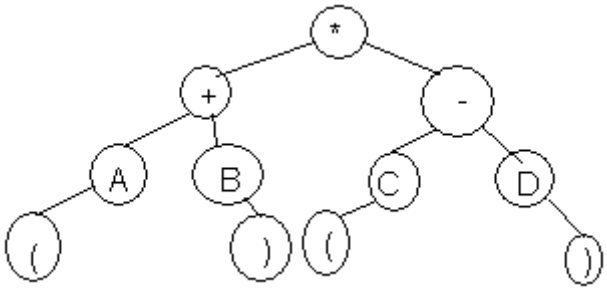
4. Inorder obilazak stabla na slici daje izraz:



RJ: (A + B) * (C – D)

---

5. Postorder obilazak stabla na slici daje izraz:



RJ: (A) B + (C) D - *

---

6. Koliko razina ima potpuno binarno stablo koje sadrži 100 čvorova i koliki je broj čvorova na posljednjoj razini ?

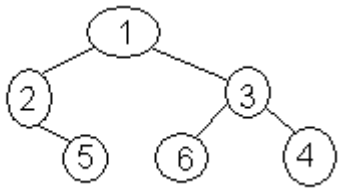
a) broj razina =6 broj čvorova=64
b) broj razina=7 broj čvorova=37 ←
c) broj razina =7 broj čvorova=50
d) broj razina =7 broj čvorova=64
e) broj razina =6 broj čvorova=50

---

7. Što će se ispisati funkcijom:

```
void ispis (cvor *korijen) {
     printf ("%c", korijen->element);
     if (korijen->lijevo && korijen->desno) {
          ispis (korijen->desno);
     ispis (korijen->lijevo);
     }
}
```

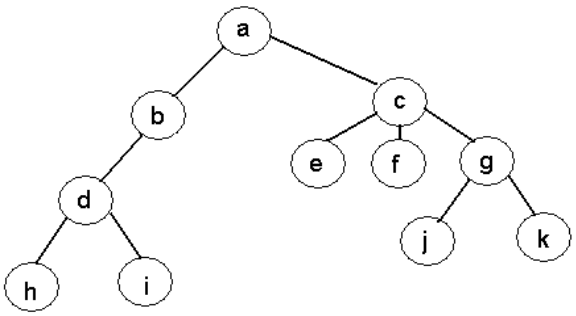
za stablo na slici pozivom funkcije?



RJ: 1 3 4 6 2

---

8. Koja od sljedećih tvrdnji nije istinita:



RJ:

a) Svi čvorovi sa stabla na slici su istog stupnja ←

b) U stablu sa slike listovi su:{h,i,e,f,j,k}

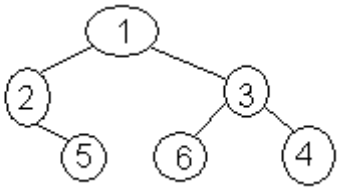
c) Maksimalni broj čvorova binarnog stabla na k-toj razini jednak je 2k-1

d) Maksimalni broj čvorova binarnog stabla dubine k jednak je 2k -1 za k>0

e) Binarno stablo koje je visine k i ima 2k -1 elemenata naziva se puno (full) binarno stablo

---

9. Zadana je neka funkcija (mislim postorder) i treba znat što ispisuje:



RJ: 4 6 3 5 2 1

---

10. Stupanj stabla (koji ima **n** razina) je:

RJ:

a) najmanji stupanj nekog čvora u stablu

b) n

c) najveći stupanj nekog čvora u stablu ←

d) broj čvorova u stablu

e) broj čvorova u potpunom stablu s *n* razina

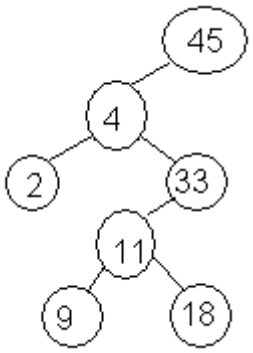
---

11. Koliko čvorova ima koso stablo sa **n** razina:

RJ:

a) 2*n-1
b) n+1
c) 2^n-1
d) n ←
e) 2^n

12.



Sortirano binarno stablo na slici (lijevo manji element, desno veći) generirano je sljedećim ulaznim nizom brojeva:

RJ: 45, 4, 33, 11, 2, 9, 18

---

13. U prazno binarno stablo uneseni su elementi 20, 15, 1, 3, 7, 48, 12, 19, 35. Kolika je dubina stabla?

RJ: 6

---

14. Što radi sljedeća funkcija:

```
void func (cvor *k, int *br){
        if (k != NULL){
              func(k->d, br);
              if(k->1 == NULL && k-> d == NULL) (*br)++;
                  func(k->d, br);
```

RJ: **a) broji listove u stablu ←**

b) broji elemente u stablu
c) računa zbroj elemenata u stablu
d) broji parne elemente u stablu
e) ništa od navedenog

---

15. Koja funkcija vraća najveći element u stablu:

```
RJ: int func(cvor *k){
            if (k->1 != NULL) return func(k->1);
                else return k->element;
```

---

16. Koja od sljedećih tvrdnji je istinita?

RJ: **a) inorder i preorder obilaskom bit će obrađeni svi čvorovi u stablu** ←

b) obilazak ppreorder moguće je jedino primijeniti na IpotpunimI stablima

c) postorder obilazak uvijek obrađuje samo listove stabla

d) preoder obilazak uvijek obrađuje samo listove stabla

e) inorder obilazak stabla obrađuje dvostruko više elemanata nego postorder obilazak

---

17. Uz prethodno ispravno deklarirane sve podatke i već formirano binarno stablo, što radi sljedeća funkcija:

```
void ispis (cvor *glava) {
      if (glava != NULL) {
            ispis (glava->lijevo);
            ispis (glava->desno);
            printf ("%s \n", glava->element);
      }
}
```

RJ:

a) uvijek ispisuje samo vrijednost elementa na koji pokazuje glava

b) funkcija ne radi ništa jer je tipa *void*

c) inorder ispisuje vrijednosti elemanata stabla

d) preorder ispisuje vrijednosti elemenata stabla

e) postorder ispisuje vrijednosti elemenata stabla ←

---

18. Što radi slijedeća funkcija?

```
int f(cvor *glava) {
      int i = 0;
      if (glava) {
            if (glava->lijevo || glava->desno) i++;
            i += f(glava->lijevo);
            i += f(glava->desno);
      }
      return i;
}
```

RJ:

a) Broji čvorove u stablu koji imaju lijevo dijete.

b) Broji čvorove u stablu koji imaju oba djeteta.

c) Broji čvorove u stablu koji imaju desno dijete.

d) Niša od navedenog.

e) Broji čvorove u stablu koji nisu listovi (imaju bar jedno dijete). ←

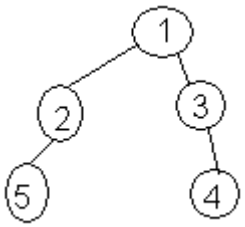
---

19. Što radi sljedeća funkcija:

```
int fx (cvor *glava) {
      if (glava) {
            return fx(glava->l) + fx(glava->d) + 1;
      } else return 0;
}
```

RJ:
a) broji razine stabla
b) računa zbroj elemenata u stablu
c) vraća vrijednost >=1 ako je stablo potpuno, 0 inače
d) broji listove stabla
e) broji čvorove stabla ←

20.



RJ: Stablo nije moguće ispisati sortirano inorder, preorder i postorder obilaskom

---

21. Pretraživanje binarnog stabla najbrže je ukoliko se radi o:

RJ: sortiranom potpunom stablu

---

22. Najefikasniji algoritam stvaranja gomile od *n* elemenata za najgori slučaj ima složenost:
RJ: a) $O(n*\log_2 n)$
b) $O(\log_2 n)$
c) $O(1)$
d) $O(n^2)$
e) O(n) ←

---

23. Koja je od sljedećih tvrdnji za gomilu točna?

RJ: Gomila se koristi kada je potrebno do najvećeg ili najmanjeg podatka doći algoritmom složenosti **O(1)**. Reorganizacija gomile nakon uklanjanja najvećeg ili najmanjeg podatka je složenosti **$O(\log_2 n)$**. Novi element u gomilu će se dodati algoritmom **$O(\log_2 n)$**.

---

24. Koji od ponuđenih ispisa gomile po razinama je ispravan ako je gomila formirana za ulazni niz 5, 10, 7, 3, 1, 90 algoritmom čija je složenost za najgori slučaj $O(n\log_2 n)$?

RJ:

a) 90
5 10
3 1 5

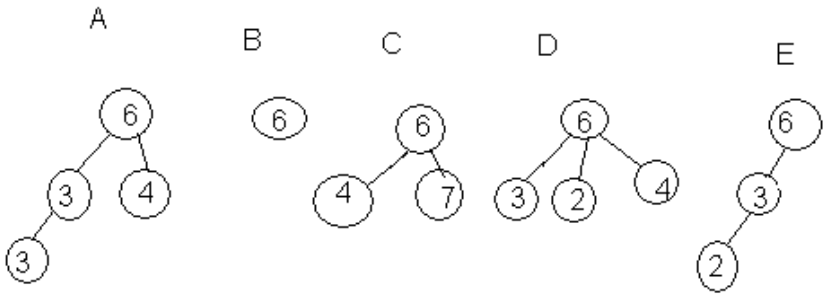
b) 90
10 7
5 1 3

c) 90 ←
5 10
3 1 7

d) 90
10 7
3 1 5

---

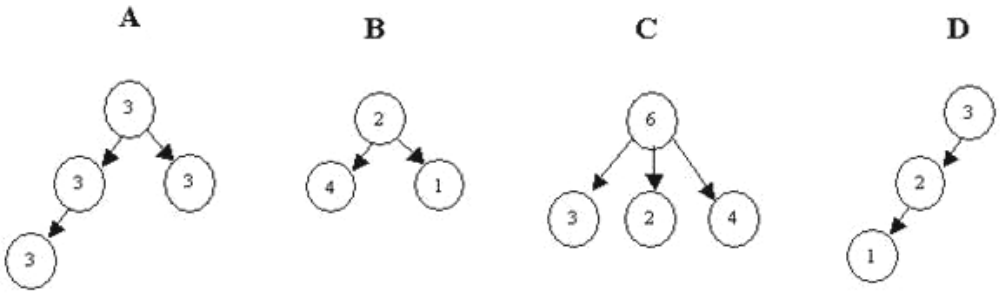
25. Koja od prikazanih stabla su gomile:



RJ: A, B

---

26. Koja od prikazanih stabala su gomile:



RJ:

a) A ←

b) B

c) C

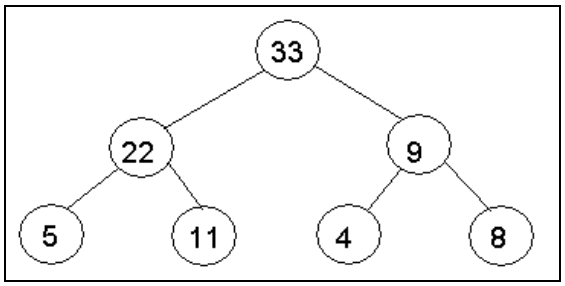
d) niti jedno od pokazanih atabala nije gomila

e) A, B, C, D

---

27. Koja slika prikazuje gomilu oblikovanu ulaznim nizom (11, 33, 4, 5, 22, 59) tako da za najgori slučaj složenost bude $O(n\log_2 n)$?

Rj:



---

28. Kada se gomila oblikuje dodavanjem jednog po jednog elementa u stablo uz očuvanje strukture gomile, tada je vrijeme izvođenja oblikovanja gomile za najgori slučaj (n je broj ulaznih elemenata):

RJ: a) O(log2n)
b) O(n)
c) O(n2 * log2n)
d) O(n * log2n) ←
e) O(n2)

29.

20 3 15 2 1 14 10

Ako je gomila realizirana u polju, u kojem od sljedećih elementi zapisani u polju:

RJ:
20
3 15
2 1 14 10

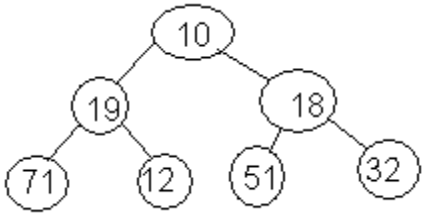
---

30. Stvori gomilu – složenost u najgorem slučaju poboljšane funkcije:

RJ: O(n)

---

31. Podaci za koje je potrebno stvoriti gomilu smješteni su u stablo na sljedeći način – koji će se prvi element zamijeniti (tu mi fali dio pitanja):

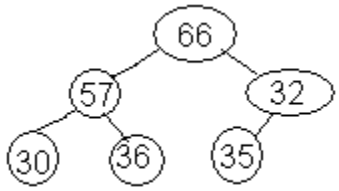


RJ: 51 i 18

---

32. 66 57 32 30 36 - zadano polje, ako se doda 35 na kraj sa kime će zamijeniti mjesto (tako nekako ide pitanje):

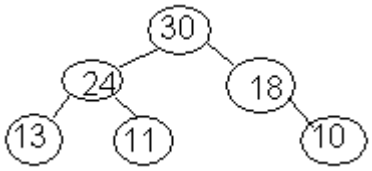
RJ: Broj 35 će zamijeniti mjesto s brojem 32



---

33. Koji od sljedećih ne zadovoljava gomilu:

RJ: 10 je na krivoj strani



---

34. Koje ne zadovoljava svojstvo gomile (ne sjećam se detalja):

RJ: 10 7 8 9 6 5 4 3 2 1

---

35. 22 33 44 99 66 77 88 55 – zadano je polje i koji element će se zamijeniti (ne sjećam se detalja):

RJ: 44 i 88

36.

RJ: Gomila je potpuno binarno stablo takvo da je podatak u nekom čvoru **<u>veći ili jednak</u>** podacima u čvorovima svoje djece.

37.

99 88 66 22 77 55 33 11 – zadana je gomila (poljem), nakon prvog podešavanja pri uzlaznom heap sortu što se dogodi (ne sjećam se točno pitanja):

RJ: 88 77 66 22 11 55 33 99

---

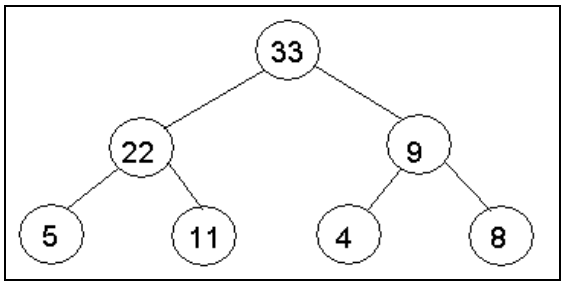
38. Struktura stog se dinamički najčešće predstavlja?

Rj: Jednostruko povezanom linearnom listom

---

39. Koja slika prikazuje gomilu oblikovanu ulaznim nizom (11, 33, 4, 5, 22, 59) tako da za najgori slučaj složenost bude $O(n\log_2 n)$?

Rj:



---

40. Pretraživanje binarnog stabla najbrže je ako se radi o:

Rj: Sortiranom potpunom stablu

---

41. Što radi:

```
void ispisi(cvor *glava) {
      cvor *p;
      for (p=glava; p!=NULL;p=p->sljed)
            printf("%d\n", p->element);
}
```

Rj: Ispisuje sve vrijednosti elemenata jednostruko povezane linearne liste.

---

42. U prazno binarno stablo uneseni su elementi 20, 15, 1, 3, 7, 48, 12, 19, 35. Kolika je dubina stabla?

Rj: 6

---

43. Koja je od slijedećih tvrdnji za gomilu točna?

Rj: Gomila se koristi kada je do najvećeg/najmanjeg potrebno doći sa složenošću $O(1)$. Složenost reorganizacije nakon uklanjanja prvog člana je $O(\log_2 n)$. Složenost dodavanja novog člana u gomilu je $O(\log_2 n)$.

---

44. Koji od ponuđenih ispisa gomile po razinama je ispravan ako je gomila formirana za ulazni niz 50 5 7 10 13 1 8 algoritmom čija je složenost za najgori slučaj O(nlog2n)?

a) 50
   13 10
   8 7 5 1
b) 50
   5 7
   10 13 1 8
c) 50
   13 7
   5 10 1 8
d) 50 ←
   13 8
   5 10 1 7
e) 50
   13 8
   10 7 5 1

---

45. Što ispisuje funkcija

```
void ispisi( struct cvor *glava ) {
      if( glava != NULL  && glava->elem % 2) {
            printf(" %d ", glava->elem);
            ispisi(glava->sljed);
       }
}
```

ako se u jednostruko povezanoj listi na koju pokazuje parametar glava nalaze sljedeći cijeli brojevi :
1 57 43 13 8 11 20 10 56 53

a) 1 57 43 13 ←
b) ne ispisuje ništa
c) 1 57 43 13 8 11
d) 1 57 43 13 8 11 20 10 56 53
e) 1 57 43 13 11 53

---

46. Ispravna deklaracija dvostruko povezane liste u memoriji glasi:

```
a) struct s1 {
       int mbr;
       char ime_pr[50];
       int spol;
       long pret;
       long sljed;
       }

b) struct s1 {
       int mbr;
       char ime_pr[50];
       int spol;
       zapis *pret;
       zapis *sljed;
        }

c) struct s1 {
       int mbr;
       char ime_pr[50];
       int spol;
       struct s1 *sljed;
       }

d) typedef struct s1{     ←
       int mbr;
       char ime_pr[50];
       int spol;
       } zapis1;

typedef struct s2{
       zapis1 element;
       struct s2 *pred;
       struct s2 *sljed;
       } zapis;

e) struct s1 {
       int mbr;
       char ime_pr[50];
       int spol;
       long *pret;
       long *sljed;
       }
```

---

47. Što će se ispisati funkcijom:

```
void ispis (cvor *korijen) {
      printf ("%c", korijen->element);
      if (korijen->lijevo && korijen->desno) {
            ispis (korijen->desno);
      ispis (korijen->lijevo);
      }
}
```

za stablo na slici pozivom funkcije

```
            1
        2     3
      5    6  4
```

ispis (korijen);
ako je korijen u trenutku poziva pokazivač na korijen stabla?

a) 134625
b) 123456
c) 521634
d) 13462 ←
e) 12364

---

48. Koji od ponuđenih ispisa gomile po razinama je ispravan ako je gomila formirana za ulazni niz 5, 10, 7, 3, 1, 90 algoritmom čija je složenost za najgori slučaj O(n)?

a) 90
7 10
3 1 5
b) 90 ←
10 7
3 1 5
c) 90
5 10
3 1 5
d) 90
10 7
5 1 3
e) 5
10 7
3 1 5

---

49. Koliko čvorova ima *koso* stablo s **n** razina?

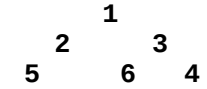
a) 2*n-1
b) n+1
c) 2^n-1
d) n ←
e) 2^n

---

50. Što će ispisati funkcija:

```
void ispis (cvor *korijen) {
      printf ("%c", korijen->element);
      if (korijen->lijevo && korijen->desno) {
            ispis (korijen->desno);
            ispis (korijen->lijevo);
      }
}
```

za stablo



RJ: 1 3 4 6 2

---

51. Kada se gomila oblikuje dodavanjem jednog po jednog elementa u stablo uz očuvanje strukture gomile, tada je vrijeme izvođenja oblikovanja gomile za najgori slučaj (n je broj ulaznih elemenata):

a) O(log2n)
b) O(n)
c) O(n2·log2n)
d) O(n·log2n) ←
e) O(n2)

---

52. Od elemenata {20, 15, 1, 3, 7, 48, 12, 19, 35} formirano je sortirano binarno stablo. Kolika je dubina stabla?
RJ: 6

---

53. Koja procedura pronalazi zadani element u jednostruko povezanoj listi?

```
a)
cvor *trazi1(cvor *glava,tip element) {
      cvor *p;
      for (p=glava;p!=NULL;p++)
            if (p->element==element)
                  return p;
      return NULL;
}

b)
cvor *trazi1(cvor *glava,tip element) {
      cvor *p;
      if (p->element==element) return p;
      return NULL;
}

c)
cvor *trazi1(cvor *glava,tip element) {
      cvor *p;
      for (p=glava;p!=NULL;p=p->sljed)
            if (p->element!=element)
                  return p;
      return NULL;
}
```

```
d)
cvor *trazi1(cvor *glava,tip element){          ←
      cvor *p;
      for (p=glava;p!=NULL;p=p->sljed)
            if (p->element==element)
                  return p;
      return NULL;
}
```

```
e)
cvor *trazi1(cvor *glava,tip element){
      cvor *p;
      for (p=glava;p!=NULL;p++)
            if (p->element!=element)
                  return p;
      return NULL;
}
```

---

54. Koji je stupanj stabla s **n** razina?

a) najmanji stupanj nekog čvora u stablu
b) n
c) najveći stupanj nekog čvora u stablu ←
d) broj čvorova u stablu
e) broj čvorova u potpunom stablu s **n** razina

---

55. Što radi zadana funkcija za poziv `f(glava,7,&br)` ako je glava pokazivač na početak jednostruko povezane liste?

```
cvor *f(cvor *p,int broj,int *br) {
      if (p) {
            ++(*br);
            if (p->broj==broj)
                  return p;
            else
                  return f(p->sljed,broj,br);
      } else return NULL;
}
```

RJ: vraća pokazivač na čvor koji sadrži `broj` i njegov redni broj u listi (preko `br`)

---

56. Koja tvrdnja **nije točna** za ovu funkciju:

```
skiniizred(tip *element,tip red[], int n, int *izlaz,int ulaz){
      if (ulaz==*izlaz) return 0;
      (*izlaz)++;
      *izlaz%=n;
      *element=red[*izlaz];
      return 1;
}
```

RJ: Funkcija vraća 1 ako se iz reda može skinuti samo 1 element

---

57. Dinamička struktura za jednostruko povezanu listu sadrži:

RJ: Pokazivac na prvi element liste i proizvoljan broj čvorova

---

58. Ovdje je napisana funkcija za dodavanje elemenata u stog ostvaren jednostruko povezanom listom. Na kraju funkcije je izostavljena jedna naredba. Treba izabrati naredbu koja ide na to mjesto da funkcija radi.
Funkcija otprilike izgleda ovako:

```
dodajelement(element **vrh,int vrijednost) {
      element novi = new element;
      novi->vrijednost=vrijednost;
      novi->slijed=*vrh;
      ##### <- Što nedostaje?
}
```

RJ: Nedostaje naredba `*vrh=novi;`

---

59. Treba li se kod funkcije dodaj_u_red (red je ostvaren listom) mijenjati i pokazivac "izlaz"?

RJ: Treba, jer ako je red prazan, moramo ga postaviti na prvi dodani element

---

60. Kako izgleda funkcija za dodavanje elemenata u red ostvaren poljem?

RJ: `int dodajured (struct zapis *red,int *ulaz,int izlaz, zapis elem, int n)`

---

61. Sto ce se dogoditi nakon sto u ovu gomilu ubacimo broj 35 i pozovemo funkciju "ubaci"?
66 57 32 30 36 ← ubacujemo broj 35

RJ: Zamijeniti ce se 35 i 32

---

62. Koja procedura trazi element u listi?

RJ:
```
cvor *trazi (cvor *g, tip elem){
      cvor *p;
      for (p=g;p!=NULL;p=p->slijed)
            if (p->vrijednost=elem) return p;
            return NULL;
}
```

---

63. Koja je slozenost funkcije ubaci_u_red za red izveden poljem
RJ: O(1)

---

64. Koji je prototip funkcije za dodavanje u red poljem. Ako je red pun, polje se dinamički udvostruči. Fja vraća 1 ako je dodavanje uspjelo, inače vraća 0.
RJ: `int DodajURed (int element, int *red, int n, int *izlaz, int *ulaz)`

---

65. Zadano je:
```
        1
      2   3
   5   9 4
```
Što će se ispisati s:
```
void ispis (cvor *korijen){
      if korijen{
            ispis (korijen->desno);
            ispis (korijen->lijevo);
            printf("%c",korijen->element);
      }
}
```

RJ: 4 9 3 5 2 1

---

66. Je li u funkciji `dodajured(int element, cvor **ulaz, cvor **izlaz)` potrebno primati pokazivač na izlaz iz reda po referenci i zašto?

RJ: Da, zato što u slučaju NULL vrijednosti glave podaci bivaju izbrisani (il neš u tom stilu)

---

67. Dinamička podatkovna struktura za real. jednostruko povezane liste sastoji se od:

RJ: pokazivača na prvi element liste i proizvoljnog broja povezanih čvorova

---

68. koja je složenost dohvaćanja zadnjeg elementa reda (red je realiziran listom)?

RJ: O(n)

---

69. Koji je prototip funkcije za skidanje elemenata iz reda listom (funkcija vraća 1 ako je uspjela skinuti element, inače vraća 0)?

RJ: `int SkiniIzReda (int *element, atom **ulaz, atom **izlaz)`

---

70. koja je složenost funkcije koja skida ZADNJI element iz reda?

RJ: O(n)

---

71. Koja tvrdnja nije ispravna za zadanu funkciju?

```
cvor *trazi(cvor *glava, int sifra){
      if (glava->sifra==sifra) return glava;
      if (glava->slijed)
            return trazi(glava->slijed, sifra);
      else
            return NULL;
}
```

RJ: F-ja nije ispravna, ne radi za praznu listu.

---

72. Sto ce se dogoditi pozivom `f(glava)`, `f(glava)` f-je:

```
void f(cvor *glava){
      if (glava->sljed){
            printf ("%s\n", glava->ime);
            f(flava->sljed);
      }
}
```

RJ: Dva puta ispisuje imena osim od posljednjeg cvora.

---

73. U red jednostruko povezanom listom pohranjuju se cjelobrojni zapisi. Prototip f-je za skidanje iz reda (1 za uspjesno, 0 neuspjesno obavljeno)

RJ: `skini(cvor **ulaz, cvor **izlaz, int *element)`

---

74. Red ostvaren ciklickim poljem, koja je tvrdnja neistinita?

```
int SkiniIzReda(tip *element, tip red[], int n, int *izlaz, int ulaz){
      if(ulaz==*izlaz) return 0;
      (*izlaz)++;
      *izlaz %=n;
      *element=red[*izlaz];
      return 1;
}
```

RJ: F-ja vraca 0, ako se iz reda moze skinuti tocno 1 element.

---

75. Koja je tvrdnja za jednostruko povezanu linearnu listu je istinita:

RJ: može ostvariti statičkom strukturom polje

---

76. Koja od ponuđenih funkcija ispravno implementira Heap sort?

```
a) void HeapSort(tip A[],int n) {
        int i;
        StvoriGomilu(A,n/2);
        for (i=n/2;i<=0;i--) {
                Zamijeni(&A[1],&A[i]);
                Podesi(A,1,i-1);
        }
}
```

```
b) void HeapSort(tip A[],int n) {
        int i;
        StvoriGomilu(A,n);
        for (i=n;i>=2;i--) {
                Zamijeni(&A[1],&A[i]);
                Podesi(A,1,i-1);
        }
}
```

←

```
c) void HeapSort(tip A[],int n) {
        int i;
        StvoriGomilu(A,1);
        for (i=1;i<=n/2;i++) {
                Zamijeni(&A[n],&A[1]);
                Podesi(A,1,i+1);
        }
}
```

```
d) void HeapSort(tip A[],int n) {
        int i;
        StvoriGomilu(A,1);
        for (i=1;i<=n;i++) {
                Zamijeni(&A[n],&A[1]);
                Podesi(A,1,i+1);
        }
}
```

```
e) void HeapSort(tip A[],int n) {
        int i;
        StvoriGomilu(A,n);
        for (i=n/2;i<=0;i--) {
                Zamijeni(&A[1],&A[i]);
                Podesi(A,1,i-1);
        }
}
```