1. Pretpostavka je da postoje funkcije za operacije nad stogom skini i dodaj sa sljedećim prototipima:

```
int skini (int *stavka, int stog[], int *vrh)
```

int dodaj (int stavka, int stog[], int n, int *vrh)

Što će ispisati sljedeći program?

```
#include <stdio.h>
#define MAXSTOG 100
int main() {
      int stog[MAXSTOG],vrh=-1;
      int a=1,b=2,c=3;
      dodaj(a,stog,MAXSTOG,&vrh);
      dodaj(b,stog,MAXSTOG,&vrh);
      dodaj(c,stog,MAXSTOG,&vrh);

      skini(&a,stog,&vrh);
      skini(&c,stog,&vrh);
      skini(&b,stog,&vrh);
      printf("%d %d %d",a,b,c);
}
```

a) 3 1 2
b) 1 2 3
c) 2 3 1
d) 1 3 2
e) 2 1 3

2. Na stog se pohranjuju samo cijeli brojevi. Prototip funkcije za skidanje cijelog broja sa stoga je (funkcija vraća 0 ili 1, ovisno o tome da li se zapis uspio skinuti s vrha stoga):

a) `int skini(int stavka, int stog[], int *vrhStog);`
b) `int skini(int *stavka, int stog[], int *vrhStog);`
c) `int skini(float stavka, float stog[], int vrhStog);`
d) `void *skini(int *stavka, int stog[], int n, int *vrhStog);`
e) `int *skini(int *stavka, int stog[], int vrhStog);`

5. Složenost funkcije

```
int dodaj (zapis stavka, zapis stog[], int n, int *vrh) {
      if (*vrh >= n-1) return 0;
      (*vrh)++;
      stog [*vrh] = stavka;
      return 1;
}
```

a) složenost ovisi o veličini zapisa stavke, pa se ne može jednoznačno odrediti
b) O(log n)
c) O(log2 n)
d) O(n)

e) O(1)

6. Kakav je sadržaj stoga nakon izvođenja funkcije funkcija, ako je stog prije poziva prazan? Funkcije za operacije nad stogom skini i dodaj vraćaju 1 ako su obavile traženu zadaću, a 0 ako nisu, te imaju sljedeće prototipe:

```
int skini (int *stavka, int stog[], int *vrh)
int dodaj (int stavka, int stog[], int n, int *vrh)
      #include <stdio.h>
      #define MAXSTOG 100
            void funkcija() {
                  int stog[MAXSTOG],
                  pomStog[MAXSTOG];
                  int i, vrh = -1, pomVrh = -1;
            while (skini(&i, stog, &vrh)) {
                  if (i>=0) dodaj(i, pomStog, MAXSTOG, &pomVrh);
            }
            while (skini(&i, pomStog, &pomVrh)) {
                  if (i<0){
                        dodaj(i, stog, MAXSTOG, &vrh);
                  }
            }
```

a) Stog sadrži samo elemente <=0

b) Stog je prazan

c) Sadržaj stoga je nepoznat

d) Sadržaj stoga je nepromijenjen

e) Stog sadrži samo elemente >0

9. Na stog se pohranjuju samo cijeli brojevi. Prototip funkcije za stavljanje cijelog broja na stog je (funkcija vraća 0 ili 1 ovisno o tome da li se zapis uspio pohraniti na vrh stoga):

a) `int dodaj(int stavka, int stog[], int n, int VrhStog);`

b) `int dodaj(float stavka, float stog[], int n, int VrhStog);`

c) `void dodaj(int stavka, int stog[], int n, int *VrhStog);`

d) `int dodaj(int stavka, int stog[], int n, int *VrhStog);`

e) `int *dodaj(int *stavka, int stog[], int n, int VrhStog);`

10. Za stog realiziran cjelobrojnim poljem postoje funkcije `push` i `pull` koje stavljaju, odnosno uzimaju element sa stoga. Ukoliko je vrh stoga na lijevoj strani, što će se nalaziti na stogu nakon izvršavanja sljedećeg programskog odsječka (na početku je stog prazan):

```
for(i=1;i<=10;i++)
      push(i);
for(j=1;j<=5;j++)
      pull();
```

a) 6 7 8 9 10

b) 1 2 3 4 5

c) 1

d) 1 2 3 4 5 6 7 8 9 10

e) neće biti više elemenata na stogu

11. Stog je struktura za koju vrijedi:
a) Da bi pristupili elementu s dna stoga, potrebno je sve ostale skinuti.
b) Zadnji element koji smo stavili na stog zadnjega ćemo i skinuti
c) Omogućava direktan pristup svakom upisanom elementu
d) FIFO
e) Ništa od navedenog

12. Ako imamo cjelobrojni stog i funkciju uzmi koja uzima element sa stoga i ima sljedeći prototip (funkcija vraća 1 ako je uspješno skinula element, a 0 ako nije):

```
int uzmi(int STOG[], int n, int *vrh, int *stavka);
```

kako bi se napisala funkcija koja računa broj elemenata na stogu

a)
```
int br_elem( int STOG[], int n, int *vrh) {
      int elem, br=0;
      while( uzmi(&STOG, n, vrh, &elem) != 0 ) br++;
      return br;
}
```
b)
```
int br_elem( int STOG[], int n, int *vrh) {
      int elem, br=0;
      while( uzmi(STOG, n, vrh, &elem) != 0 ) br++;
      return br;
}
```
c)
```
int br_elem( int STOG[], int n, int *vrh) {
      int elem, br=0;
      while( uzmi(&STOG[0], n, vrh, elem) != 0 ) br++;
      return br;
}
```
d)
```
int br_elem( int STOG[], int n, int *vrh) {
      int elem, br=0;
      while( uzmi(STOG, n, *vrh, &elem) != 0 ) br++;
      return br;
}
```
e)
```
int br_elem( int STOG[], int n, int *vrh) {
      int elem, br=0;
      while( uzmi(STOG, n, vrh, elem) != 0 ) br++;
      return br;
}
```

13. Najefikasniji algoritam stvaranja gomile od `n` elemenata za najgori slučaj ima složenost:
a) $O(n \cdot \log_2 n)$
b) $O(\log_2 n)$
c) $O(1)$
d) $O(n_2)$
e) O(n)

14. Stupanj stabla (koji ima n razina) je:
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 sa n razina

18. Koliko čvorova ima koso stablo s n razina?

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

19. 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 $O(n\log_2 n)$?

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

e) 90
10 3
7 5 1

20. 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

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

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

22. Što radi sljedeća funkcija:

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

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

24. Ispravna deklaracija dvostruko povezane liste u memoriji glasi:

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;
```

26. 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;
    }
```

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).

28. 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
5 10
3 1 5

b) 90
10 7
3 1 5

c) 90
7 10
3 1 5

d) 90
10 7
5 1 3

e)90
10 7
3 1 5

30. 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);
            }
      }
```

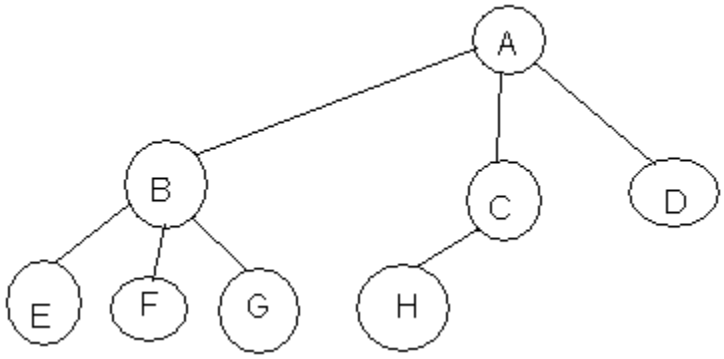
e)
```
void HeapSort (tip A[], int n) {
      int i;
      StvoriGomilu (A, n);
      for (i = n/2; i <=0 ; i--) {
            Zamijeni (&A[1], &A[i]);
            Podesi (A, 1, i-1);
            }
      }
```

31. Pretraživanje binarnog stabla **najbrže** je ako se radi o: **Sortiranom potpunom stablu**

34. Koja je od slijedećih tvrdnji za gomilu točna?

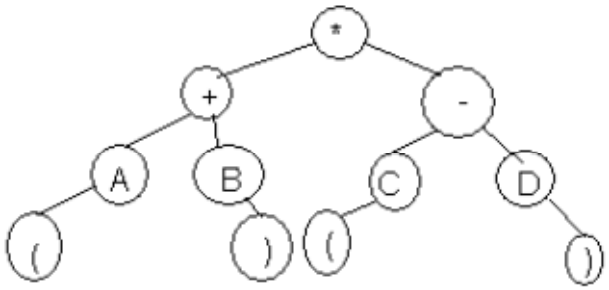
RJ:
- Gomila se koristi kada je do najvećeg/najmanjeg potrebno doći sa složenošću *O(1)*.
-Složenost reorganizacije nakon uklanjanja prvog člana je *O(log$_2$n).*
-Složenost dodavanja novog člana u gomilu je O*(log$_2$n).*

35. Stablo ima sljedeće atribute:



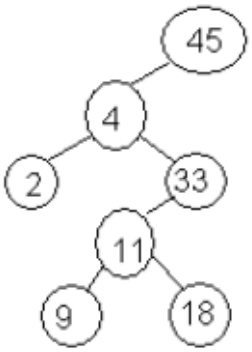
RJ: stupanj = 3, dubina = 3, potpuno

36. Inorder obilazak stabla na slici daje izraz1:
Postorder obilazak stabla na slici daje izraz2:



RJ1: (A + B) * (C – D)
RJ2: (A) B + (C) D - *

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



a) 45, 4, 2, 9, 11, 33, 18
b) 45, 4, 33, 11, 2, 9, 18
c) 45, 9, 11, 18, 2, 33, 4
d) 9, 11, 18, 2, 33, 4, 45

42. Ako je stog realiziran cjelobrojnim poljem od n elemenata, kolika je apriorna složenost skidanja SVIH elemenata sa stoga:
a) O(1)
b) O(n)
c) $O(n_2)$
d) $O(\log_2 n)$
e) ovisi o operacijskom sustavu

44. Ukoliko je ulaz = 1, a izlaz=4, koliko ima elemenata u redu realiziranom pomoću cirkularnog polja, ako je veličina polja 10 (pretpostavite da ulaz pokazuje na prvi prazan element, dok izlaz pokazuje na prvi stavljeni element)?

a) 7

b) 6

c) 3

d) 5

e) ne može se odrediti

45. Neka je na sljedeći način napisana funkcija koja skida element tipa `tip` iz reda realiziranog cikličkim poljem:

```
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;
      }
```

Koja je od sljedećih tvrdnji **lažna**?

a) Složenost funkcije je O(1).

b) Funkcija vraća 0, ako se iz reda može skinuti točno jedan element.

c) Za poziv funkcije, kada u redu postoji barem jedan zapis koji se može skinuti iz reda, funkcija vraća 1.

d) Funkcija vraća 1, ako je zapis uspješno skinut iz reda.

e) Za poziv funkcije, kada je red prazan, funkcija vraća 0.

46. U red realiziran jednostruko povezanom listom pohranjuju se zapisi koji sadrže cijele brojeve. Prototip funkcije za skidanje zapisa iz tako realiziranog reda je (funkcija vraća 1 ili 0, ovisno o tome je li zapis uspješno skinut iz reda):

a) `int skini (cvor **ulaz, cvor **izlaz, int element);`

b) `int skini (cvor **ulaz, cvor **izlaz, int *element);`

c) `void skini (cvor **ulaz, cvor **izlaz, int *element);`

d) `int skini (cvor *ulaz, cvor **izlaz, int element);`

e) `int skini (cvor *ulaz, cvor **izlaz, int *element);`

49. U dvostruko povezanu listu spremaju se zapisi slijedećeg tipa:

```
typedef struct s1{
      int mbr; // matični broj studenta
      char ime[40+1]; // ime studenta
      float prosjek; // prosjek ocjena
      struct s1 *sljed;
      struct s1 *preth;
} zapis;
```

Kako glasi funkcija koja izbacuje prvi element iz liste, oslobađa zauzetu memoriju te vraća 1 ako je operacija uspjela, a 0 ako operacija nije uspjela?

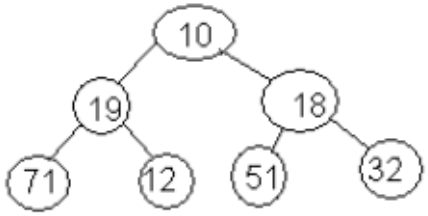
a)

```
int izbaci(zapis **glava, zapis **rep) {
      if (*glava == NULL) return 0;
      if ((*glava)->sljed == NULL) {
```

```
            free(*glava);
            *glava = *rep = NULL;
      }
      else {
            *glava = (*glava)->sljed;
            free((*glava)->preth);
            (*glava)->preth = NULL;
      }
      return 1;
}
```

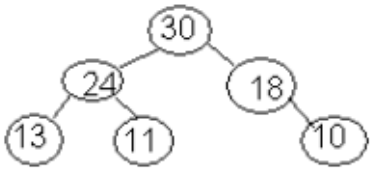
51. Stvori gomilu – složenost u najgorem slučaju poboljšane funkcije:
RJ: O(n)

52. 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

53. Koji od sljedećih ne zadovoljava gomilu:



RJ: 10 je na krivoj strani

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

55. Dinamička struktura za jednostruko povezanu listu sadrži:
RJ: Pokazivac na prvi element liste i proizvoljan broj čvorova

56. 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;`

58.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

59. Kako izgleda funkcija za dodavanje elemenata u red ostvaren poljem?
RJ: `int dodajured (struct zapis *red,int *ulaz,int izlaz, zapis elem, int n)`

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

61. 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)

62. koja je složenost dohvaćanja zadnjeg elementa reda (red je realiziran listom)?
RJ: O(n)

63. 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)`

64. koja je složenost funkcije koja skida ZADNJI element iz reda?
RJ: O(n)

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

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

RJ: Dva puta ispisuje imena osim od posljednjeg cvora.

66. 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)`

67. 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)`