ASP (FER1)

Primjeri zadataka sa 1. blica

bliceve skupio i uredio: Tomislav

1. Kako treba glasiti naredba `printf` za kod programa da se ispise `2 3 1` ?

```
char a,b;
char *p1,*p2;
a='2'; b='3';
p1=&a; p2=&b;
```

Rješenje: `printf("%c %c %d",*p1,*p2,*p2-*p1);`

2. Koja od sljedećih naredbi ispravno provjerava da li je veza programa i datoteke ispravno prekinuta?

RJ: `if(fclose(f) = = 0){ printf("Uspješno zatvaranje"); }`

3. Koje će vrijednosti poprimiti varijable a i b nakon izvršenja programa?

```
void main(){
      int a=10,b=2,*p;
      p=&a;
      b=*p;
      b=a;
}
```

RJ: `a=10 b=10`

4. Koju vrijednost poprima varijabla c nakon izvršenja programa?

```
int a=1,b=2;  float c=2.5;
c=fun(&a,&b);
```

funkcija fun izgleda ovako:

```
fun(int *a,int *b){
      b=a;
      return *b/*a;
}
```

RJ: `1.`

5. Što će se ispisati?

```
void f(int x,int *y){
      x%=2;
      *y * x;
}
void main(){
      int a=8,b=2;
      printf("%d %d",a,b);
      f(a,&b);
      printf("->%d %d",a,b);
}
```

RJ: `8 10 -> 8 10`

6. Što će se ispisati?

```
void f(int x,int *y){
      x%=2;
      *y*=x;
}
void main(){
      int a=8,b=2;
      printf("%d %d",a,b);
      f(a,&b);
      printf("->%d %d",a,b);
}
```

RJ: `8 10 -> 8 0`

7. Nakon izvršavanja naredbe `fprintf`, što će se nalaziti zapisano u datoteci (gledajući binarno) ?

```
fprintf(fp, "%c\n", '0' );
```

RJ: `00110000 00001010`

8. Što će se ispisati sljedećim programom ?

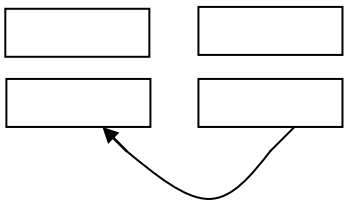
```
void f( int a ) {
  a++;
  printf("a=%d ", a);
}
void main() {
  int a=0, b=2;
  f(a++);
  f(++a);
  f(b);
}
```

RJ: `133`

9. Adresa pohranjena u pokazivaču deklariranom kao `int *p;` povećat će se za `1` bajt sljedećom naredbom:

RJ: `p=(int *)((char *)p + 1);`

10. Uz pretpostavku da je veličina kućice proporcionalna zauzeću memorije, kojem programskom odsječku odgovara sljedeći slijed slika?



a)
```
int a, *b;
b = *a;
```
b)
```
int *a, b;
b = &a;
```
c)
```
int *a, b;
a = &b;
```
d)
```
int a, *b;
b = &a;
```
e)
```
int a, *b;
b = a;
```

Rješenje: D

11. Što će se ispisati programom:

```
int fun (char *c, int i) {
  (*c)++; i--;
  return (*c) * i;
}
main () {
  char c = '0'; int i = 1, j;
  j = fun (&c, i);
  printf ("%d %d %d", c, i, j);
}
```

a) 49 1 0
b) 48 1 0
c) Program je sintaktički neispravan
d) 48 0 0
e) 49 0 0

Rješenje: A

12. Što se ispiše na zaslonu kao rezultat izvođenja sljedećeg programskog odsječka:

```
char f1(int i) {
  i += 1;
  return i;
}
void f2(int i) {
  i += 1;
}
void main() {
  int i = 1;
  i=f1(i);
  printf("%d,",i);
  f2(i);
  printf("%d", i);
}
```

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

Rješenje: B

13. Što će se ispisati izvođenjem sljedećeg odsječka?

```
char p, char r, char *pp, char *pr, char *pom;
p='p';pp=&p;
r='r';pr=&r;
pom=pp;
pp=pr;
pr=pom;
p='r';r='p';
printf("%c %c %c %c",r,*pp,p,*pr);
```

a) p r r r
b) r p p r
c) p p r r
d) r p r p
e) r r p p

Rješenje: C

14. Što će biti zapisano u datoteku sljedećim programskim odsječkom?

```
int i = 4; FILE *datIzlaz;
fprintf(datIzlaz, "%3d", i);
```

a) 00000000 00000000 00110100
b) 00000100
c) 00000000 00000000 00000100
d) 00100000 00100000 00110100
e) 01100100

Rješenje: D

15. Što će se ispisati sljedećim programskim odsječkom ?

```
float p[3]={3,2,1};
float *pp;
pp=p;
printf("%f", *(pp+1));
```

a) 1
b) 2
c) 3
d) 5
e) 4

Rješenje: B

16. Što će se ispisati izvršavanjem sljedećeg programa ?

```
void main() {
  int  a, b, *p;
  a=10; b=5; p=&a;
  *p = b++;
  printf("a=%d b=%d", a, b);
}
```

a) a=5 b=5
b) a=11 b=6
c) a=5 b=6
d) a=10 b=6
e) a=11 b=5

Rješenje: C

17. Što će se ispisati sljedećim programom?

```
#include <stdio.h>
void uvecaj(int a, int *b) {
    a = a + 1; *b = *b + 1;
}
void main() {
    int a = 0, b = 0;
    printf("a=%d b=%d  ", a, b);
    uvecaj(a, &b);
    printf("a=%d b=%d\n", a, b);
  }
```

a) Pogreška–funkcija koja mijenja parametre ne smije biti tipa `void`
b) Pogreška–varijable a i b imaju nedefiniranu vrijednost u funkciji
c) a=0 b=0 a=0 b=1
d) a=0 b=1 a=1 b=2
e) a=1 b=1 a=0 b=0

Rješenje: C

18. Što će se ispisati na zaslon sljedećim programskim odsječkom, ako datoteka `test.dat` ne postoji na magnetskom disku?

```
FILE *f;
f = fopen ("test.dat", "r");
if (f) {
    printf ("Datoteka već postoji");
} else {
    printf ("Datoteka ne postoji");
}
```

a) Program će dojaviti pogrešku jer varijabla `f` treba biti tipa `int`
b) Varijabli tipa `FILE *` ne može se pridružiti vrijednost na opisani način
c) Neće se ispisati ništa, jer će operacijski sustav dojaviti pogrešku
d) Datoteka već postoji
e) Datoteka ne postoji

Rješenje: E

19. Nakon izvršavanja naredbe `fprintf`, što će se nalaziti zapisano u datoteci (gledajući binarno) ?

```
fprintf(fp, "%c\n", '0' );
```

a) `00110000`
b) `00110000 00001010`
c) `00110000 00000000`
d) `00000000 00001010`
e) `00000000`

Rješenje: B

20. Adresu pohranjena u pokazivaču `int *p` povećati za `1 byte`

Rješenje: `p=(int *) ((char *)p+1);`

21. Što će ispisati:

```
int f(int i) {
      i /=2;
      printf (« %d», i);
      return i;
}
void main () {
      int i=5, j;
      j=f(i);
      printf (« %d %d», i, j);
```

Rješenje: `2 5 2`

22. Što će ispisati:

```
int f(int *a, int *b) {
      int x, y;
      x=(*a)++;
      y=(*b)++;
return x+y;
}
void main () {
      int c=1, d=1, rez;
      rez=f(&c, &d);
      printf («c=%d d=%d rez=%d», c, d, rez);
}
```

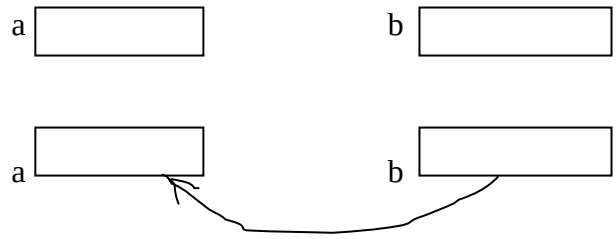
Rješenje: `2 2 2`

23. Što će se pohraniti u varijablu a slijedećim prog. odsječkom

```
short *a;
short a1=1;
a=&a1
```

Rješenje: adresa varijable `a1`

24. Uz pretpostavku da je veličina kučice propocionalna zauzeću memorije, kojem prog. odsječku odgovara slijedeća slika



Rješenje

```
int a, *b;
b=&a;
```

25. Kojom od naredbi se otvara datoteka za čitanje i pisanje na kraj datoteke uz očuvanje starog sadržaja

Rješenje `f=fopen(« dat.txt «, «a+»);`

26. Što će biti zapisano u neformatiranu datoteku slijedećom naredbom

```
char c='A';
File *datoteka;
datoteka = fopen («test.dat», «wb»);
fwrite (&c, sizeof (c), 1, datoteka);
```

Rješenje: `01000001`

27. Koliki je `x i y` nakon poziva funkcije

```
void f(int *x, int *y) {
     int z=*x;
     *x = *y;
     *y = z;
}
void main () {
     int x=5, y=6;
     f(&x, &y);
```

Rješenje `x=6, y=5`

28. Što će se ispisati

```
int x, y, *p, *q;
x=2, y=3;
p=&x;
q=p;
y=*q;
printf (« %d», y);
```

Rješenje `2`

29. Kao bi izgledao sadržaj datoteke nakon izvršenja slijedećeg prog. odsječka

```
struct nesto {
     short i;
     char c[2];
} s;
FILE *d;
s.i=7;
strcpy (s.c, «77»);
fwrite (&s, sizeof (s), 1, d);
```

Rješenje: `00000000 00000111 00110111 00110111`

30. Što će ispisati sljedeći programski odsječak:

```
void zam(int *a, int*b){
     *a=*b;
     *b=*a;
}
void main(){
     int x=1, y=2;
     zam(&x,&y);
     printf(" %d %d ",x,y);
     zam(&y,&x);
     printf(" %d %d ",x,y);
}
```

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

Rješenje: C

31. Što će ispisati slijedeći odsječak koda:

```
int a=3, b=4, *c;
a += b;
c = &b;
*c %= ++a;
c = &a;
printf("%d %d %d", a,b,*c);
```

a) `8 4 4`
b) `8 4 8`
c) `8 8 4`
d) `8 0 8`
e) `8 0 0`

Rješenje: B

32. Koja situacija u memoriji odgovara sljedećem programskom odsječku (veličina kućice je proporcionalna broju bajtova koji zauzima podatak) ?

```
char a, *b;
b = &a;
```

RJ: veća(b) pokazuje u manju(a)

33. Što će ispisati sljedeći odsječak koda:

```
void f(int *x, int *y) {
     int *pom;
     pom = x; x = y; y = pom;
     }
void main() {
     int x=2, y=3;
     f(&x, &y);
     printf("%d %d", x, y);
}
```

a) `2 3`
b) Ne može se odrediti
c) `1 2 3 4 5`
d) `3 2`
e) U programu je sintaksna pogreška

Rješenje: A

34. Što će se ispisati sljedećim programom ?

```
void f( int a ) {
       a++;
       printf("a=%d ", a);
}
void main() {
       int a=0, b=2;
       f(a++);
       f(++a);
       f(b);
}
```

a) `a=1 a=3 b=1`
b) `a=1 a=2 b=1`
c) `a=1 a=4 a=2`
d) `a=1 a=3 a=3`
e) `a=1 a=4 b=2`

Rješenje: A

35. Na disku postoji datoteka `"tu.sam";` naredbama

```
FILE *fp;
fp = fopen("tu.sam","a");
```

otvorit će se datoteka :

a) za pisanje s tim da će se pokazivač postaviti na kraj datoteke.
b) za čitanje i pisanje s tim da se pokazivač postavi na kraj datoteke.
c) za pisanje s tim da će se prebrisati postojeća datoteka.
d) za čitanje i pisanje s tim da se pokazivač postavi na početak datoteke.
e) samo za čitanje.

Rješenje: B

36. Adresa pohranjena u pokazivaču deklariranom kao `int *p;` povećat će se za 1 bajt sljedećom naredbom:

a) `p = *p + 1;`
b) `*p = *p + 1;`
c) `p = (int *) ((char *) p +1);`
d) `*p = (int *) ((char *) *p +1);`
e) `p = p + 1;`

Rješenje: A

37. Koja od sljedećih tvrdnji nije istinita, ako imamo naredbu `*p=7;`

a) pokazivač p mora prethodno biti inicijaliziran da bi naredba bila logički ispravna.

b) Nova adresa na koju pokazuje pokazivač `p` nakon naredbe

```
p = p + broj;
```

može se dobiti na sljedeći način:

```
nova_adresa = stara_adresa + broj*sizeof(long);
```

c) Ako je deklarirano polje `int a[5]` sljedeće naredbe, kojima pristupamo trećem elementu polja, su ekvivalentne:

```
*(a+2)
a[2]
```

d) Pokazivače se može uspoređivati.

e) Pokazivaču se može oduzeti i dodati cijeli broj.

Rješenje: A

38. Što će se ispisati sljedećim programom?

```
#include <stdio.h>
  void main() {
    int a = 2, b, *p;
    p = &b;
    b = 0;
    *p = a;
    printf("%d, %d\n", a, b);
  }
```

a) `2, 2`

b) `0, 2`

c) `2, 0`

d) `0, 0`

e) Pogreška : neispravno inicijalizirana vrijednost pokazivača `p`

Rješenje: B

39. Što će se ispisati sljedećim programskim odsječkom?

```
void f(int a, int *b){
     a=*b;
     *b=a+1;
}
void main(){
     int a=1, b=2;
     f(a,&a);
     printf(" %d%d",a,b);
     f(a,&a);
     printf(" %d%d",a,b);
}
```

Rješenje: `22 23`

40. p je definirano kao `int *p`. Kako ćemo adresu pohranjenu u `p` uvećati za jedan `byte`?

RJ: `p=(int*)((char*)p+1);`

41. Što će biti zapisano u datoteci?

```
int i=2; FILE *datIzlaz;
fprintf(datIzlaz, "%2d",i);
```

RJ: `0010000 00110010 (32 50)`

42. Što će ispisati sljedeći programski odsječak:

```
void zam(int *a, int*b){
     *a=*b;
     *b=*a;
}
void main(){
     int x=1, y=2;
     zam(&x,&y);
     printf(" %d %d ",x,y);
     zam(&y,&x);
     printf(" %d %d ",x,y);
}
```

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

Rješenje: C

43. Što će ispisati slijedeći odsječak koda:

```
int a=3, b=4, *c;
a += b;
c = &b;
*c %= ++a;
c = &a;
printf("%d %d %d", a,b,*c);
```

a) 8 4 4
b) 8 4 8
c) 8 8 4
d) 8 0 8
e) 8 0 0

Rješenje: B

44. Što će ispisati sljedeći odsječak koda:

```
void f(int *x, int *y) {
  int *pom;
  pom = x; x = y; y = pom;
}
void main() {
  int x=2, y=3;
  f(&x, &y);
  printf("%d %d", x, y);
}
```

a) 2 3
b) Ne može se odrediti
c) 1 2 3 4 5
d) 3 2
e) U programu je sintaksna pogreška

Rješenje: A

45. Što će se ispisati sljedećim programom ?

```
void f( int a ) {
  a++;
  printf("a=%d ", a);
}
void main() {
  int a=0, b=2;
  f(a++);
  f(++a);
  f(b);
}
```

a) a=1 a=3 b=1
b) a=1 a=2 b=1
c) a=1 a=4 a=2
d) a=1 a=3 a=3
e) a=1 a=4 b=2

Rješenje: A

46. Na disku postoji datoteka "tu.sam"; naredbama

```
FILE *fp;
fp = fopen("tu.sam","a");
```

otvorit će se datoteka :

a) za pisanje s tim da će se pokazivač postaviti na kraj datoteke.
b) za čitanje i pisanje s tim da se pokazivač postavi na kraj datoteke.
c) za pisanje s tim da će se prebrisati postojeća datoteka.
d) za čitanje i pisanje s tim da se pokazivač postavi na početak datoteke.
e) samo za čitanje.

Rješenje: B

47. Što će se ispisati sljedećim programom:

```
#include <stdio.h>
  void main() {
    int a = 2, b, *p;
    p = &b;
    b = 0;
    *p = a;
    printf("%d, %d\n", a, b);
  }
```

a) 2, 2
b) 0, 2
c) 2, 0
d) 0, 0
e) Pogreška : neispravno inicijalizirana vrijednost pokazivača p

Rješenje: B

48. Adresa pohranjena u pokazivaču deklariranom kao int *p; povećat će se za 1 bajt sljedećom naredbom:

a) p = *p + 1;
b) *p = *p + 1;
c) p = (int *) ((char *) p +1);
d) *p = (int *) ((char *) *p +1);
e) p = p + 1;

Rješenje: A

49. Što će ispisati sljedeći programski odsječak:

```
void zam(int *a, int*b){
     *a=*b;
     *b=*a;
}
void main(){
     int x=1, y=2;
     zam(&x,&y);
     printf(" %d %d ",x,y);
     zam(&y,&x);
     printf(" %d %d ",x,y);
}
```

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

Rješenje: B

50. Što će ispisati sljedeći programski odsječak:

```
void f(int x){
    x++;
    printf(" x=%d",x);
}
void main(){
    int x=1;
    f(x);
    f(x);
    f(x);
    printf(" x=%d",x);
}
```

a) x=2 x=2 x=2 x=1
b) x=2 x=3 x=4 x=4
c) x=1 x=2 x=3 x=4
d) x=1 x=2 x=3 x=3
e) x=2 x=3 x=4 x=1

Rješenje: A

51. Koja će od sljedećih naredbi pohraniti vrijednost varijable b u varijablu a?

```
int a = 2, b = 3;
```

a) `a = *(&b);`
b) `*a = &b;`
c) `*a = *b;`
d) `&a = *b;`
e) `a == b;`

Rješenje: A

52. Kojom od naredbi otvaramo datoteku iz koje nije dopušteno čitanje:

```
FILE *f;
```

a) `f = fopen("osobe.txt", "a+");`
b) `f = fopen("osobe.txt", "a");`
c) `f = fopen("osobe.txt", "w+");`
d) `f = fopen("osobe.txt", "r+");`
e) `f = fopen("osobe.txt", "r");`

Rješenje: B

53. Ukoliko prva `printf` funkcija iz priloženog programskog odsječka ispiše:

```
1245032 2.710000
```

što će ispisati sljedeća `printf` funkcija ?
(pretpostavka je da se memorija adresira sa 4 bajta, dakle `unsigned long`)

```
void main(){
     float *pok, var[2]={2.71, 3.14};
     pok=var;
     printf("\n%lu %f",pok,*pok);
     pok++;
     printf("\n%lu %f",pok,*pok);
}
```

a) `1245033 2.710000`
b) `1245034 3.140000`
c) `1245036 2.710000`
d) `1245036 3.140000`
e) `1245034 2.710000`

Rješenje: D

54. Što će se ispisati slijedećim programom?

```
int f (int *a, int b) {
  *a = *a + b;
  b = *a * b;
  return *a + b;
}
void main () {
  int a = 1, b = 2, c = 3;
  c += f (&a, b);
  printf ("%d %d %d\n", a, b, c);
}
```

a) `1 2 12`
b) `3 6 12`
c) `3 6 9`
d) `3 2 9`
e) `3 2 12`

Rješenje: E

55. Što će biti zapisano u datoteku sljedećim programskim odsječkom?

```
…
int i = 2; FILE *datIzlaz;
…
fprintf(datIzlaz, "%2d", i);
…
```

a) `00000000 00110010`
b) `01100010`
c) `00000000 00000010`
d) `00000010`
e) `00100000 00110010`

Rješenje: E

56. Što će se ispisati izvršavanjem sljedećih naredbi:

```
int i = 1;
int j = 10;
int *p = &j;
i += j;
p = &i;
printf("%d, %d, %d", i, j, *p);
```

a) `11, 11, 11`
b) Pogreška : pokazivači ne mogu mijenjati vrijednost
c) `10, 10, 11`
d) Svaki put će ispisati drugu vrijednost za `*p`
e) `11, 10, 11`

Rješenje: E

57. Što će se ispisati sljedećim programom?

```
#include <stdio.h>
  void ByValue(int a, int b) {
    int p = a; a = b; b = a;
  }
  void main() {
    int a = 2, b = 3;
    printf("a=%d b=%d  ", a, b);
    ByValue(a, b);
    printf("a=%d b=%d\n", a, b);
  }
```

a) Pogreška–datoteka s zaglavljem zove se `studio.h` a ne `stdio.h`
b) `a=2 b=3  a=2 b=3`
c) `a=3 b=3  a=3 b=3`
d) Neće se ispisati ništa
e) `a=2 b=2  a=2 b=2`

Rješenje: B

ASP (FER1)
Primjeri zadataka sa 2. blica

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

Rješenje: A

2. Zadane su međusobno rekurzivne funkcije:

```
ispisl(char *niz, int n) {
  if ( (*niz != '\0') && (n >= 0) ) {
    printf("%c", *niz);
    ispisd(niz+1, n-1);
  }
}
ispisd(char *niz, int n) {
  if ( (*(niz + n) != '\0') && (n >= 0) ) {
    printf("%c", *(niz+n));
    ispisl(niz, n-1);
  }
}
```

Što će se ispisati slijedećim pozivom funkcije?

```
ispisl("ABECEDA", 5);
```

a) ABECEDA
b) ABECE
c) ADBEEC
d) CEEBDA
e) ABECED

Rješenje: E (?)

3. Što će ispisati sljedeći program:

```
#include <stdio.h>
void r(int n) {
    if (n<=3) return;
    printf("%2d", n);
    r(n-1);
    r(n-2);
    printf("%2d", n);
    return;
}
main() {
    r(6);
}
```

a) 6 5 6 5 5 4 4 6
b) 6 5 4 4 5 4 4 6
c) 6 6 5 5 4 4
d) 6 5 4 5 4 4
e) neće ispisati ništa

Rješenje: B

4. Koja od slijedećih tvrdnji je istinita za dovoljno velik broj n?
   a) O(n2) < O(2n) < O(n3)
   b) O(n2) < O(n3) < O(2n)
   c) O(n3) < O(n2) < O(n)
   d) O(1) < O(n) < O(log2n)
   e) O(2n) < O(n3) < O(3n)

Rješenje: D

5. 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);`

Rješenje: B

6. Koja od sljedećih nizova naredbi u pseudokodu će zamijeniti vrijednost varijabli A i B pomoću stoga:
   a) `stavi(A); skini(B); stavi(B); skini(A);`
   b) `stavi(B); skini(A);`
   c) `stavi(A); skini(B);`
   d) `stavi(A); stavi(Pom); stavi(B); stavi(Pom); skini(A); stavi(Pom);skini(B);`
   e) `stavi(A); stavi(B); skini(A); skini(B);`

Rješenje: E

7. Koja od navedenih tvrdnji je istinita?
   a) O(2n)>O(n2)
   b) O(n2)>O(2n)
   c) O(n!)<O(2n)
   d) O(n2)<O(nlog2n)
   e) O(2n2+n)=O(n2)

Rješenje: E

8. Što radi sljedeća funkcija?

```
long f(int poc, int kraj){
  if (poc<=kraj){
    if (poc%2==1)
      return poc + f(poc+1,kraj);
    else
      f(poc+1,kraj);
  }
  else
    return 0;
}
```

a) Sumira sve parne brojeve u intervalu `[poc,kraj]`
b) Sumira sve prim brojeve u intervalu `[poc,kraj]`
c) Sumira sve brojeve u intervalu `[poc,kraj]`
d) Uvijek vraća vrijednost 0 bez obzira na raspon intervala `[poc,kraj]`
e) Sumira sve neparne brojeve u intervalu `[poc,kraj]`

Rješenje: E

9. Koja od ponuđenih funkcija računa sumu `n` članova sljedećeg reda:

a)
```
float red(int n){
  if(n<=1) return 1.;
  else
    return (n/2?1.:-1.)/(2*n-1);
}
```
b)
```
float red(int n){
  if(n<=1) return 1.;
  else
    return (n/2?1.:-1.)/(2*n-1);
}
```
c)
```
float red(int n){
  if(n<=1) return 1.;
  else
    return (n%2?1.:-1.)/(2*n-1) + red(n-1);
}
```
d)
```
float red(int n){
  if(n<=1) return 1.;
  else
    return 2*red(n)-1;
}
```
e)
```
float red(int n){
  if(n<=1) return 1.;
  else
    return (n%2?1.:-1.)/(2*red(n)-1);
}
```

Rješenje: ?

10. Koja od navedenih tvrdnji je istinita?

a) `O(2n)>O(n2)`
b) `O(n2)>O(2n)`
c) `O(n!)<O(2n)`
d) `O(n2)<O(nlog2n)`
e) `O(2n2+n)=O(n2)`

Rješenje: E

11. Imate dvije funkcije `func1` i `func2` definirane na sljedeći način:

```
void func1( int n1, int n2) {
    int  i;
    for( i=n1; i<=n2; i++ ) printf("\n %d", i);
}
void func2( int n1, int n2 ) {
    if( n1 <= n2 ) {
        printf("\n %d", n1);
        func2(n1 + 1, n2);
      }
}
```

Koja od sljedećih tvrdnji nije istinita ?

a) Funkcija `func2` je složenosti `O(n log2n)`
b) Funkcija `func1` će se brže izvršiti
c) Funkcije `func1` i `func2` imaju istu složenost
d) Rezultat izvođenja funkcija je jednak za iste parametre `n1` i `n2`
e) Funkcija `func1` je složenosti `O(n)`

Rješenje: ?

12. Koje je apriorno vrijeme f-je koja vraća najmanji el. matrice

```
int min(int *mat, int n){
  int i, min;
  min =*mat;
  for (i=1;i<n*n;i++){
    if (min>*(mat+1)){
      min=*(mat+i);
    }
  }
  return min;
}
```

Rj: O(n2)

13. Koja od slijedećih tvrdnji je istinita za dovoljno velik broj `n`?

a) `O(n2) < O(2n) < O(n3)`
b) `O(n2) < O(n3) < O(2n)`
c) `O(n3) < O(n2) < O(n)`
d) `O(1) < O(n) < O(log2n)`
e) `O(2n) < O(n3) < O(3n)`

Rješenje: D

14. Koja od sljedećih tvrdnji je istinita, za dovoljan velik n:

a) O(1) < O(log(n)) < O(n) < O(nlog(n)) < O(n2) < O(n3) < O(2n)
b) O(1) < O(log(n)) < O(nlog(n)) < O(n) < O(n2) < O(n3) < O(2n)
c) O(logn) < O(1) < O(nlog(n)) < O(n) < O(n2) < O(n3) < O(2n)
d) O(logn) < O(1) < O(n) < O(nlogn) < O(n2) < O(n3) < O(2n)
e) O(1) < O(log(n)) < O(n) < O(nlog(n)) < O(2n) < O(n2)< O(n3)

Rješenje: A

15. Ako `push` stavlja na `stog` i vraća `1` za uspješno stavljanje a `0` za neuspješno, te `pop` skida sa stoga i vraća skinuti element za uspješno skidanje ili -1 za neuspješno odredi sto će se nalaziti ne stogu.

```
push(5);
push(push(pop()));
```

RJ: 51

16. Što radi sljedeća funkcija?

```
long f(int poc, int kraj){
if (poc<=kraj){
 if (poc%2==1)
return poc + f(poc+1,kraj);
else       f(poc+1,kraj);
 }
else    return 0;}
```

a) Sumira sve parne brojeve u intervalu `[poc,kraj]`
b) Sumira sve prim brojeve u intervalu `[poc,kraj]`
c) Sumira sve brojeve u intervalu `[poc,kraj]`
d) Uvijek vraća vrijednost 0 bez obzira na raspon intervala `[poc,kraj]`
e) Sumira sve neparne brojeve u intervalu `[poc,kraj]`

Rješenje: E

17. 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`

Rješenje: A

18. Programski odsječak

```
for (i = 0; i < 1; i++)
for (j = 0; j < 1; j++)     ...
```

ima apriornu složenost:
a) O(i)
b) O(1)
c) O(n)
d) ne može se odrediti bez izvođenja na računalu
e) O(j)

Rješenje: B

19. Odredi O-notaciju.

```
int n;
...
while (n>0) {
      n/=3;
}
```

RJ: $O(\log_3 n)$

20. Što se ispisuje sljedećim programom?

```
int f(int n) {
  if(n <= 0) return 0;
  if(n%2) return 1 + f(n-1);
  return n + f(n-1);
}
void main() {
  printf("%d",f(5));
}
```

a) Greška : rekurzija se beskonačno poziva
b) 5
c) 15
d) 11
e) 9

Rješenje: E

21. Fibonnaccievi brojevi (rekurzija):

```
int f(int n){
  if(n<=1) return 1;
  else return F(n-2)+F(n-1);
}
```

22. Koju će vrijednost vratiti funkcija `func` ako je se pozove s `func(polje,3,2);` a polje je deklarirano kao

```
polje[3][3]={1,2,3,1,2,3,1,2,3};
int func (int *p, int n, int i){
     int pom;
     pom = p[i]+p[n-i];
     if (i <= 0) return pom;
     else return pom + func (p, n, i-1);
}
```

a) 6
b) 9
c) 12
d) 10
e) 0

Rješenje: C

23. Odredi sto je istina:

a) `O(2n)>O(pow(n,2))`
b) `O(n!)<O(pow(n,2))`
c) `O(2*pow(n,2)+n)=O(pow(n,2))`

Rješenje: C

24. Što se događa?

```
void f(int n){
      if(n==2)return;
      if(n==5)return;
      f(n-2);
       printf ("%d",n);
 }
void main() {
  f(9);
getch ();
}        //pise 79
```

25. Asimptotsko vrijeme izvođenja sljedeće rekurzivne funkcije:

```
void pisi2(int *p, int i, int n) {
if (i<n) {
      pisi2 (p,i+1,n);
      printf ("%d ", *(p+i));
      }
}
```

a) o(n*log n)
b) o(n2)
c) o(n)
d) o(2n)
e) o(4n)

Rješenje: C

26. Koju će vrijednost vratiti funkcija `func` ako je se pozove s `func(polje,3,2);` a polje je deklarirano kao

```
polje[3][3]={1,2,3,1,2,3,1,2,3};
int func (int *p, int n, int i){
      int pom;
      pom = p[i]+p[n-i];
      if (i <= 0) return pom;
      else  return pom + func (p, n, i-1);
}
```

a) 6
b) 9
c) 12
d) 10
e) 0

Rješenje: C

27. 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)

Rješenje: E

28. Što se ispisuje nakon izvođenja sljedećeg programskog odsječka:

```
void f(int *p, int n, int *max, int *i) {
     if(n == -1) return;
     if(p[n] > *max ) {
     *max = p[n];
     }
f(p,n-1,max,i);
if(p[n] == *max) {
     (*i)++;
     }
}
void main() {
     int p[] = {1,10,3,4,10};
     int max = p[0],i=0;
     f(p, 4, &max,&i);
     printf("%d, ",i);
}...
```

a) `1`
b) Greška : rekurzija nikad neće završiti.
c) `2`
d) Greška : rekurzivna funkcija ne može biti tipa `void`
e) `0`

Rješenje: C

29. Koja od sljedećih tvrdnji vezanih uz stog je istinita?
    a) interni stog računala koristi se samo pri deklaraciji globalnih varijabli C programa
    b) pojedina operacija dodaj (`push`) i brisi (`pop`) zahtijeva jednako vremena bez obzira na broj pohranjenih podataka
    c) stog je programska struktura u koju se dodaju i brišu elementi po načelu FIFO (First In First Out)
    d) za programsku realizaciju stoga moguće je isključivo koristiti stog koji je definiran kao polje
    e) na stog se mogu pohraniti isljučivo cjelobrojni (`int`) podaci

Rješenje: B

30. Što radi sljedeća funkcija?

```
long f(int poc, int kraj){
     if (poc<=kraj){
          if (poc%2==1) return poc + f(poc+1,kraj);
     else f(poc+1,kraj);
     }
else return 0;
}
```

a) Sumira sve neparne brojeve u intervalu `[poc,kraj]`
b) Sumira sve prim brojeve u intervalu `[poc,kraj]`
c) Sumira sve parne brojeve u intervalu `[poc,kraj]`
d) Sumira sve brojeve u intervalu `[poc,kraj]`
e) Uvijek vraća vrijednost 0 bez obzira na raspon intervala `[poc,kraj]`

Rješenje: A

31. 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

Rješenje: B

32. Zadane su međusobno rekurzivne funkcije:

```
ispisl(char *niz, int n) {
      if ( (*niz != '\0') && (n >= 0) ) {
            printf("%d", *niz);
            ispisl(niz+1, n-1)
      }
}
ispisd(char *niz, int n) {
      if ( (*(niz + n) != '\0') && (n >= 0) ) {
            printf("%d", *(niz+n));
            ispisl(niz, n-1);
            }
 }
```

Što će se ispisati slijedećim pozivom funkcije?

```
ispisl("ABECEDA", 5);
```

a) ADBEEC
b) ABECED
c) ABECE
d) ABECEDA
e) CEEBDA

Rješenje: B

33. Odrediti najbolji slučaj asimptotskog vremena

```
int postoji (int polje[], int n, int br){
  int i;
  for (i=0;i<n;i++){
    if (polje[i]==br) return 1;
  }
  return 0;
}
```

Rj: Za najbolji slučaj asimptotsko vrijeme izvođenja je O(1)

34.

```
int stavi (int polje[], int *vrh, int n, int element);
int skinii (int polje[], int *vrh, int element);
```

vraćaju 1 za uspješno inače 0.
Što će se ispisati sljedećim:

```
void func(){
  int i;
  int polje[10];
  int vrh;
  for(i=0;i<10;i++){
    stavi(polje, &vrh, 5, i);
  }
  while (skini(polje,&vrh,&i)){
    printf("%d",i);
  }
}
```

Rj: e) `4 3 2 1 0`

35. Što nije istina?
Rj: Rekurzivni programi su brži

36.

```
float (f(int n)){
 if(n<=0) return 0;
 else return 1./(i*i)+f(n-1);
}
```

Rj: Odgovara za `f(n)={ 0 za n<=0; 1/i(2) + f(n-1) za n>=1`

37. Složenost f-je za `n!=0`

```
void pisi(int *p, int n){
  while(n!=0){
    pisi(p+1,n-1);
    printf("%d",*p);
  }
}
```

Rj: O(oo)

38. Koje je apriorno vrijeme f-je koja vraća najmanji el. matrice

```
int min(int *mat, int n){
  int i, min;
  min =*mat;
  for (i=1;i<n*n;i++){
    if (min>*(mat+1)){
      min=*(mat+i);
    }
  }
  return min;
}
```

Rj: O(n2)

39. Što je istinito?
    Rj: `O(2n2+n)=O(n2)`

40. Funkcija stavljanja na stog (`int(push(int element)`) vraća 1 za uspjeh, 0-neuspjeh, funkcija skidanja (`int(pop())` vraća vrijednost elementa s vrha ili -1 ako je stog prazan. Što će biti?
    `pop(push(push(pop()))` (stog je bio prazan)

    Rj: b)`-1`

41. Ako imamo cjelobrojni stog i funkciju koja uzima element sa stoga:
```
int uzmi(int stog[], int n, int *vrh, int *stavka);
```
    koja vraća 1 za uspjeh inače 0.
    Kako bi se napisala funkcija koja vrača broj el. na stogu?

    Rj:
```
int br_elem(int stog[], int n, int *vrh){
  int elem_br=0;
  while(uzmi(stog, n, vrh, &elem)!=0) br ++;
}
return br;
```

42. 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);`

    Rješenje: D

43. Dana je funkcija:
```
void func(char *niz){
     if (*niz!='\0'){
        func(niz+1);
        printf("%c",*niz);
     }
}
```
    Što će se ispisati sljedećim pozivom funkcije:
    `func("ABECEDA");`
    a) `ABECEDA`
    b) Funkcija neće nikad završiti. Neće se ispisati ništa.
    c) `A`
    d) `ADECEBA`
    e) Funkcija nikad neće završiti. Slovo `A` ispisat će se "beskonačno" puta.

    Rješenje: D

44. Odredi što je istina:
    a) O(2n)>O(pow(n,2))
    b) O(n!)<O(pow(n,2))
    c) O(2*pow(n,2)+n)=O(pow(n,2))
    ...
    i još neke

    Rješenje: C

45. Koja je složenost sljedeće funkcije?

```
void pisi(int *p, int n){
     while (n!=0){
      pisi(p+1,n-1);
      printf("%d",*p);
     }
}
```

a) O(n2)
b) O(2n)
c) O(beskonačno)
d) O(n)
e) O(3n)

Rješenje: D (?)

46. 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

Rješenje: A

47. Pod pojmom rekurzije podrazumijeva se:
a) potprogram mora pozvati sistemsku funkciju time
b) u program se mora uključiti datoteka s zaglavljem rec.h
c) potprogram se mora zvati recursion
d) potprogram ne prima nikakve ulazne parametre
e) potprogram poziva sam sebe uz konačan broj poziva

Rješenje: E

48. Apriorna složenost funkcije

```
int fakt(int n){
       if (n <= 1) return 1;
       else  return n * fakt(n-1);
}
```

a) O(n)
b) O((n-1)!)
c) O(n!)
d) O(n2)
e) O(1)

Rješenje: A

49. 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

Rješenje: A

50. 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;
    }
```

Rješenje: B (ili možda E)

51. Apriorna složenost funkcije

```
long pot(long x, long y) {
   if (y <= 0) return 1;
   else return x * pot(x, y - 1);
  }
```

a) O(x)
b) O(y2)
c) O(1)
d) O(x*y)
e) O(y)

Rješenje: E

52. Koja od navedenih tvrdnji je istinita?

a) O(n^2)>O(2^n)
b) O(n!)<O(2^n)
c) O(n^2)<O(nlog2n)
d) O(2^n)>O(n^2)
e) O(2n^2+n)=O(n^2)

Rješenje: E

53. Ako funkcija stavljanja na stog vraća 1 u slučaju uspjeha a 0 u slučaju neuspjeha i ima prototip

```
int push (int element);
```

a funkcija skidanja sa stoga vraća vrijednost element s vrha ili -1 ako je stog prazan i ima prototip

```
int pop ();
```

Što će biti na stogu nakon obavljanja sljedećih naredbi, uz pretpostavku da je stog bio prazan i da stog raste s lijeva na desno:

```
push(push(pop()));
pop();
```

a) Stog će biti prazan.
b) `-1 0`
c) `1`
d) `-1`
e) `1 -1`

Rješenje: D

54. Ako `push` stavlja na stog i vraća 1 za uspješno stavljanje te 0 za neuspješno, a `pull` skida sa stoga i vraća vrijednost skinutog elementa te se uzima da ne postoji slučaj da `pull` ne uspije skinuti sa stoga odredi što će se nalaziti na stogu.

```
push(push(push(5))+pull());
```

55. Definicija nekurzivne funkcije.

56. Definicija rekurzivne funkcije.

57. Dva primjera rekurzije neki programi pa treba odredi što će se dogodit (prilično jednostavno ☺).

ASP (FER1)
Primjeri zadataka sa 3. blica

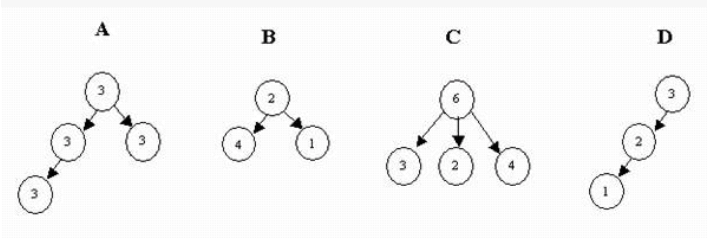
1. Najefikasniji algoritam stvaranja gomile od `n` elemenata za najgori slučaj ima složenost:
   a) $O(n \cdot \log_2 n)$
   b) $O(\log_2 n)$
   c) $O(1)$
   d) $O(n^2)$
   e) $O(n)$

   Rješenje: E

2. 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

   Rješenje: C

3. Koja od prikazanih stabala su gomile:



   a) A
   b) B
   c) C
   d) niti jedno od prikazanih stabala nije gomila
   e) A, B, C, D

   Rješenje: A

4. Uz prethodno 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);
   }
}
```

   a) uvijek ispisuje samo vrijednost elementa na koji pokazuje glava
   b) funkcija ne radi ništa jer je tipa void
   c) inorder ispisuje vrijednost elementa stabla
   d) preorder ispisuje vrijednost elementa stabla
   e) postorder ispisuje vrijednost elementa stabla

   Rješenje: E

5. Koja od sljedećih tvrdnji je istinita:
   a) `inorder` i `preorder` obilaskom bit će obrađeni svi čvorovi u stablu
   b) obilazak `preorder` moguće je jedino primijeniti na *punim* stablima
   c) `postorder` obilazak uvijek obrađuje samo listove stabla
   d) `preorder` obilazak uvije obrađuje samo listove stabla
   e) `inorder` obilazak stabla obrađuje dvostruko više elemenata nego postorder obilazak

   Rješenje: A

6. Koliko čvorova ima koso stablo s n razina?
   a) 2*n - 1
   b) n + 1
   c) 2^n - 1
   d) n
   e) 2^n

   Rješenje: D

7. 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

   Rješenje: C

8. 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

   Rješenje: A

9. Koji od sljedećih algoritama za sortiranje je najmanje uputno koristiti za sortiranje polja sa velikim brojem zapisa?
   a) Quick sort
   b) Bubble sort
   c) Merge sort
   d) Heap sort
   e) Shell sort

   Rješenje: B

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

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

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

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

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

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;
    }
```

Rješenje: E

11. 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

Rješenje: E

12. Kod Quick sorta, najbolje je za stožer odabrati?

a) slučajni odabir je najbolje
b) svejedno je kako se bira stožer
c) element srednji po vrijednosti u polju
d) zadnji element polja
e) prvi element polja

Rješenje: C

13. Neka jednostruko povezana lista sadrži čvorove sa sljedećim tipom zapisa:

```
struct s {
     int broj;
     struct s *sljed;
};
typdef struct s cvor;
```

Što radi funkcija f, ako je poziv funkcije f(glava, 7, &br)?

```
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;
     }
}
```

Napomena: varijabla `glava` je pokazivač na prvi čvor u listi, a varijabla `br` je deklarirana i inicijalizirana kao `int br = 0;`

a) Funkcija vraća pokazivač na prvi čvor u listi, te broj čvorova u listi koji sadrže broj 7
b) Funkcija vraća pokazivač na početni čvor u listi, te broj čvorova u listi koji sadrži broj 7
c) ??
d) ?? Ne vidi se na screenshotu ☹
e) ??

Rješenje: D

14. Vrijeme izvođenja sorta umetanjem je:

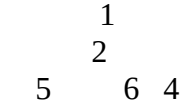
a) O(n3)
b) O(n * log2n)
c) O(n2 * log2n)
d) O(n2)
e) O(n)

Rješenje: D

15. 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



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

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

Rješenje: D

16. 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;
      }
```

Rješenje: D

17. Koliko iznose prosječna vremena izvođenja Merge i Quick sorta?
    a) O(nlog2n) i O(log2n)
    b) O(nlog2n) i O(n)
    c) O(nlog2n) i O(nlog2n)
    d) O(nlog2n) i O(n)
    e) O(log2n) i O(log2n)

Rješenje: ?

18. 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)

Rješenje: D

19. 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).

Rješenje: E

20. 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

Rješenje: B

21. 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

Rješenje: ?

22. Koja od sljedećih tvrdnji NIJE istinita?
Slika (odnosi se samo na dva ponuđena odgovora):

NEMA SLIKE (stablo, nije puno)

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

Rješenje: ?

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

NEMA SLIKE

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
e) 2, 4, 9, 11, 18, 33, 45

Rješenje: ?

24. Što će vratiti priložena funkcija za zadanu jednostruko povezanu linearnu listu:
Slika (unutar čvorova prikazan je vrijednost varijable element):

```
int f(cvor *glava){
  if (glava){
    if (glava->element > 3)
      return glava->element + f(glava->sljed);
    else
      return f(glava->sljed);
  }else{
    return 0;
  }
}
```

a) `6`
b) `13`
c) `0`
d) `19`
e) `16`

Rješenje: ?

25. 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`

Rješenje: ?

26. 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) 5
   10 7
   3 1 5

Rješenje: ?

27. Zadano je prvih nekoliko koraka sorta ubacivanjem (polje se sortira uzlazno slijeva na desno):

12 5 9 88 23 41 4 13

5 12 9 88 23 41 4 13

5 9 12 88 23 41 4 13

5 9 12 23 88 41 4 13

Kako će izgledati polje nakon slijedeće zamjene dvaju elemenata tijekom sortiranja ubacivanjem?

a) 5 9 12 23 4 41 88 13
b) 5 9 12 23 41 88 4 13
c) 5 9 12 23 88 41 13 4
d) 5 9 12 23 88 4 41 13
e) 5 9 12 23 13 41 4 88

Rješenje: ?

28. Koji sort ima najveće memorijske zahtjeve?

RJ: Merge Sort

29. Struktura stog se dinamički najčešće predstavlja?

RJ: Jednostruko povezanom linearnom listom

30. Zadano polje brojeva

66	88	99	22	77	55	33	11

sortira se Shell Sortom sa korakom k=3. Nakon koraka polje izgleda:

22	11	55	33	77	99	66	88

31. 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);
      }
    }
```

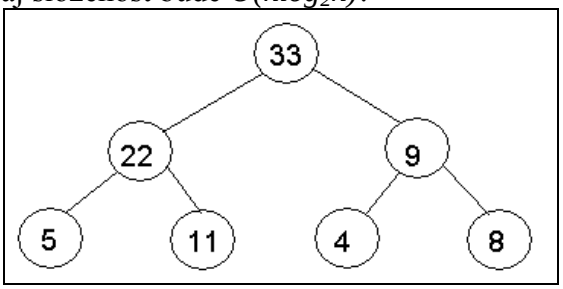
d)
```
void HeapSort (tip A[], int n) {
      int i;
      StvoriGomilu (A, 1);
      for (i = 1; i <= n; i++) {
        Zamijeni (&A[n], &A[1]);
        Podesi (A, 1, i+1);
      }
    }
```

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);
      }
    }
```

Rješenje: ?

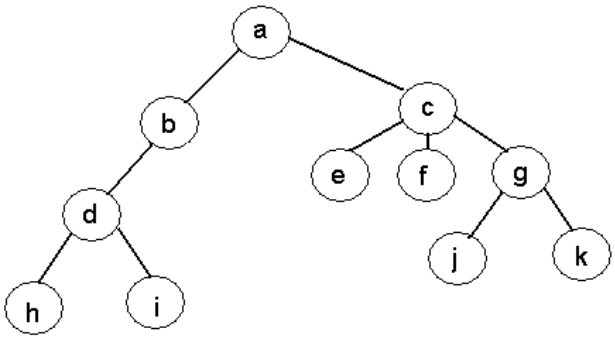
32. Koja slika prikazuje gomilu oblikovanu ulaznim nizom (11, 33, 4, 5, 22, 59) tako da za najgori slučaj složenost bude $O(n\log_2 n)$?

RJ:



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

34. Koja od slijedećih tvrdnji NIJE istinita (vezano uz sliku)?



RJ: Svi čvorovi na slici su istog stupnja.

35. 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.

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

RJ: 6

37. 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 $O(\log_2 n)$.

38. Koja je od slijedećih tvrdnji za sortove istinita?

RJ: Najgori slučaj vremena izvođenja Shell Sorta je $O(n^2)$

39. Koji od sljedećih algoritama za sortiranje je najmanje uputno koristiti za sortiranje polja sa velikim brojem zapisa?
    a) Quick sort
    b) Bubble sort
    c) Merge sort
    d) Heap sort
    e) Shell sort

Rješenje: B

40. Koji sort ima najveće memorijske zahtjeve?

RJ: Merge Sort

41. **bubble sort** - O(n^2)

42. **Sort umetanjem** - bavlja se n-1 prolaza kroz polje.
U prolazu i, i=1,...,n-1 postiže se uređenost prvih i+1 elemenata
tako da se na pravoj poziciji napravi slobodno mjesto za element s indeksom i.
Vrijeme izvođenja je O(n^2).

43. **heap sort** - element s vrha gomile zamjenjuje se s posljednjim elementom polja,
gomila se skraćuje za 1 element i podešava. Složenost podešavanja je O(log2n).
To se obavlja n puta pa je složenost sorta O(nlog2n).

44. **shell sort** - O(n2).

45. **merge sort** - Na temelju dva sortirana polja (A i B) puni se treće (C).
Koristi se strategija "podijeli pa vladaj" uz rekurziju. O(n log2 n)

46. **quick sort** - Prosječno vrijeme O(n log2 n) ,najgore O(n^2)
Ako je broj članova polja S jednak 0 ili 1, povratak u pozivni program.
Odabrati bilo koji član v u polju S. To je stožer.
Podijeli preostale članove polja S, S \ {v} u dva odvojena skupa:
S1 = { x E S \ {v} l x <= v} i S2 = { x e S \ {v} l x >= v}
Vrati {quicksort (S1), v, quicksort (S2)}

47. Kod Quick sorta, najbolje je za stožer odabrati?
a) slučajni odabir je najbolje
b) svejedno je kako se bira stožer
c) element srednji po vrijednosti u polju
d) zadnji element polja
e) prvi element polja

Rješenje: C

48. Zadano polje brojeva
66 88 99 22 77 55 33 11
sortira se Shell Sortom sa korakom k=3. Nakon koraka polje izgleda:
22 11 55 33 77 99 66 88

49. Koja je od slijedećih tvrdnji za gomilu točna?
Gomila se koristi kada je do najvećeg/najmanjeg potrebno doći sa složenošću O(1).
Složenost reorganizacije nakon uklanjanja prvog člana je O(log2n).
Složenost dodavanja novog člana u gomilu je O(log2n).

Rješenje: ?