

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

```
int dodaj (int element, Stog *stog);
```

što će biti na stogu nakon obavljanja sljedeće naredbe, uz pretpostavku da je stog bio prazan, da na njega stane najmanje 3 elementa i da stog raste s lijeva na desno?

```
dodaj(dodaj(dodaj(5, &stog)+2, &stog), &stog);
```

a)	5 3 4
<b>b)</b>	5 3 1
c)	1 1 1
d)	5 1 1
e)	1 3 5

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

```
int dodaj (int element, Stog *stog);
```

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

```
int skini (int *element, Stog *stog);
```

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

```
dodaj (5, &stog);  
  
dodaj (dodaj(skini(&element, &stog), &stog), &stog);
```

a)	5
b)	stog će biti prazan
c)	5 0
d)	-1 1
<b>e)</b>	5 1

Struktura stog kao **dinamička struktura** najčešće se predstavlja:

<b>a)</b>	Jednostruko povezanom linearnom listom
b)	Struktura stog ne može se predstaviti dinamičkom strukturom
c)	Dvostruko povezanom linearnom listom
d)	Dinamičkom strukturom gomila
e)	Binarnim stablom

Što radi sljedeća funkcija ?

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

- |    |                                 |
|----|---------------------------------|
| a) | broji listove u stablu          |
| b) | računa zbroj elemenata u stablu |
| c) | broji elemente stabla           |
| d) | broji parne elemente u stablu   |
| e) | ništa od navedenoga             |

Uz prethodno ispravno deklarirane sve podatke i već formiranu jednostruku povezanu linearnu listu, što radi sljedeća funkcija:

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

- |    |  |
|----|--|
| a) | ispisuje vrijednost svakog drugog elementa jednostruko povezane linearne liste |
| b) | funkcija ne radi ništa jer je tipa void  |
| c) | ispisuje vrijednosti elemenata svih čvorova binarnog stabla                    |
| d) | ispisuje vrijednosti parnih elemenata jednostruko povezane linearne liste      |
| e) | ispisuje vrijednosti svih elemenata jednostruko povezane linearne liste        |

Koja funkcija vraća najveći element u stablu (koje je uređeno po principu lijevo veći a desno manji) ? U stablu se nalaze cijeli brojevi.

- |    |  |
|----|--|
| a) | <pre>int func( cvor *k ) {<br/>    while( k-&gt;d != NULL ) k = k-&gt;d;<br/>    return k-&gt;elem;<br/>}</pre>            |
| b) | <pre>int func( cvor *k ) {<br/>    func(k-&gt;l);<br/>    func(k-&gt;d);<br/>}</pre>                                       |
| c) | <pre>int func( cvor *k ) {<br/>    func(k-&gt;d);<br/>    func(k-&gt;l);<br/>}</pre>                                       |
| d) | <pre>int func( cvor *k ) {<br/>    while( k-&gt;l != NULL ) k = k-&gt;d;<br/>    return k-&gt;elem;<br/>}</pre>            |
| e) | <pre>int func( cvor *k ) {<br/>    if( k-&gt;l != NULL ) return func(k-&gt;l);<br/>    else return k-&gt;elem;<br/>}</pre> |

Koja od sljedeći tvrdnji je istinita?

a)	inorder obilazak stabla obrađuje dvostruko više elemenata nego postorder obilazak
b)	postorder obilazak uvijek obrađuje samo listove stabla
c)	inorder i preorder obilaskom bit će obrađeni svi čvorovi u stablu
d)	obilazak preorder moguće je jedino primijeniti na <i>punim</i> stablima
e)	preorder obilazak uvijek obrađuje samo listove stabla

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);  
    }  
}
```

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

Neka se u gomili koja je pohranjena u polje nalaze sljedeći podaci:

66	57	32	30	36	
----	----	----	----	----	--

Što će se desiti kada se u takvu gomilu doda podatak 35 (a nakon što se pozove funkcija *ubaci* koja dodaje elemente u gomilu)?

a)	Broj 35 će zamijeniti mjesto s brojem 32
b)	Taj će se broj dodati na kraj polja jer je time očuvano svojstvo gomile
c)	Izbaci se broj 66, a 35 se umetne na to mjesto
d)	Broj 35 će zamijeniti mjesto s brojem 30
e)	Broj 35 će zamijeniti mjesto s brojem 36

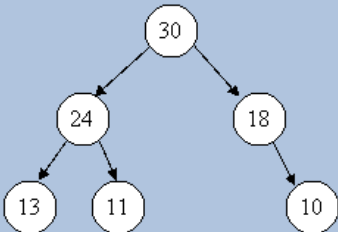

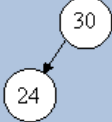
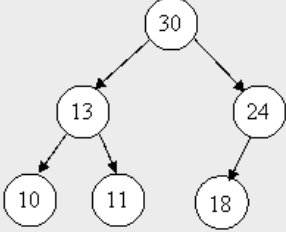
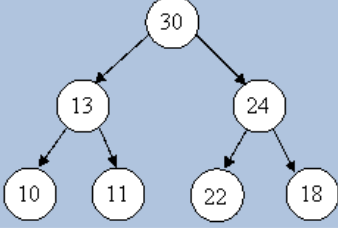
U jednostruko povezanu listu spremaju se zapisi sljedećeg tipa:

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

Kako glasi funkcija koja pronalazi u listi zapis o studentu sa zadanim matičnim brojem i vraća taj zapis u glavni program. Ako takav student ne postoji u listi funkcija vraća NULL. Lista nije sortirana.

a)	<pre>zapis *nadji(zapis *glava, int p_mbr) {     if (glava == NULL) return NULL;     if (glava-&gt;mbr == p_mbr) return glava;     else glava = glava-&gt;sljed; }</pre>
b)	<pre>zapis *nadji(zapis *glava, int p_mbr) {     while (glava &amp;&amp; (glava-&gt;mbr &lt; p_mbr))         glava = glava-&gt;sljed;     return glava; }</pre>
c)	<pre>zapis *nadji(zapis *glava, int p_mbr) {     while (glava &amp;&amp; (glava-&gt;mbr != p_mbr))         glava = glava-&gt;sljed;     return glava; }</pre>
d)	<pre>void nadji(zapis *glava, int p_mbr, zapis element) {     while (glava &amp;&amp; (glava-&gt;mbr != p_mbr))         glava = glava-&gt;sljed;     element = *glava; }</pre>
e)	<pre>void nadji(zapis *glava, int p_mbr, zapis element) {     while (glava &amp;&amp; (glava-&gt;mbr != p_mbr))         glava = glava-&gt;sljed; }</pre>

Koje od sljedećih binarnih stabala NE zadovoljava svojstva gomile?

a)	
b)	
c)	
d)	
e)	

Ako je gomila realizirana u polju, u kojem od slijedećih slučajeva elementi zapisani u polju NE zadovoljavaju svojstva gomile?	
a)	10 9 8 7 6 5 4 3 2 1
b)	10 8 9 4 5 6 7 1 2 3
c)	10 8 9 6 7 4 5 2 3 1
d)	10 7 8 9 6 5 4 3 2 1
e)	10 9 8 4 5 6 7 1 2 3

Ostvareni broj bodova: 0,50

Ako fukcija stavljanja na stog vraća 1 u slučaju uspjeha a 0 u slučaju neuspjeha i ima prototip int dodaj (int element, Stog *stog); a funkcija skidanja sa stoga vraća vrijednost elementa s vrha ili -1 ako je stog prazan i ima prototip int skini (int *element, Stog *stog); što će biti na stogu nakon obavljanja sljedeće naredbe, uz pretpostavku da je stog bio prazan i da stog raste s lijeva na desno: dodaj(dodaj(skini(&element, &stog), &stog), &stog);	
a)	1
b)	1 1
<input checked="" type="checkbox"/> c)	-1 1
d)	0
e)	stog će biti prazan

Ako je zadana struktura:  #define MAXSTOG 10  typedef struct { int vrh, polje[MAXSTOG]; } Stog; i funkcija stavljanja na stog: int dodaj (int element, Stog *stog) { if (stog->vrh >= MAXSTOG-1) return 0; stog->vrh++; stog->polje[stog->vrh] = element; return 1; } Na početku programa je: stog->vrh = -1; Koliko se ukupno elemenata može pohraniti na stog?	
a)	8 elemenata
b)	9 elemenata
c)	5 elemenata
<input checked="" type="checkbox"/> d)	10 elemenata

Zadana je struktura:

```
#define MAXSTOG 5
typedef struct {
    int vrh, polje[MAXSTOG];
} Stog;
```

Kako glasi prototip funkcije za stavljanje cijelog broja na stog (funkcija vraća 0 ili 1, ovisno o tome da li se zapis uspio pohraniti na vrh stoga):

a) void dodaj (int element, Stog stog);

b)

☒ c) int dodaj (int element, Stog \*stog);

Koja od sljedećih nizova naredbi u pseudokodu će zamijeniti vrijednost varijabli A i B pomoću stoga:

a) stavi(A); stavi(Pom); stavi(B); stavi(Pom); skini(A); stavi(Pom); skini(B);

b) stavi(B); skini(A);

c) stavi(A); skini(B);

☒ d) stavi(A); stavi(B); skini(A); skini(B);

e) stavi(A); skini(B); stavi(B); skini(A);

Funkcija za dodavanje elementa na stoga realiziran listom glasi:

☒ a)

```
int dodaj (int element, Stog *stog) {
    atom *novi;
    if ((novi = (atom*) malloc(sizeof(atom))) != NULL) {
        novi->element = element;
        novi->sljed = stog->vrh;
        stog->vrh = novi;
        return 1;
    }
    else return 0;
}
```

b)

```
int dodaj (int element, Stog *stog) {
    atom *novi;
    if ((novi = (atom*) malloc(sizeof(atom))) != NULL) {
        novi = element;
        return 1;
    }
    else return 0;
}
```

```
int dodaj (int element, Stog *stog) {
    atom *novi;
    if ((novi = (atom*) malloc(sizeof(atom))) != NULL) {
        novi->element = stog->vrh;
```

Funkcija push stavlja elemente na stog. Ako je operacija uspješno obavljena funkcija vraća 1, a u slučaju greške vraća 0. Prototip funkcije je:

```
int push (int element, Stog *stog);
```

Funkcija pop skida element sa stoga i vraća njegovu vrijednost ili -1 u slučaju greške. Prototip funkcije pop je:

```
int pop (Stog *stog);
```

Što će ispisati sljedeći programski odsječak, uz pretpostavku da je prije izvođenja stog prazan i da na njemu ima dovoljno mjesta.

```
for (i=0; i<5; i++)
    push(i, &stog);

for (i=5; i>=0; i--)
    printf("%d ", pop(&stog));
```

- a) 0 1 2 3 4
- b) 5 4 3 2 1 0
- c) 4 3 2 1 0
- d) 4 3 2 1 0 -1**
- e) 0 1 2 3 4 5

Predložena je funkcija za pronalazak čvora sa zadanom cjelobrojnom šifrom u jednostruko povezanoj linearnoj listi:

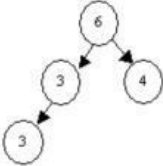
```
cvor *trazi(cvor *glava, int sifra){
    if (glava->sifra == sifra) return glava;
    if (glava->sljed)
        return trazi(glava->sljed, sifra);
    else
        return NULL;
}
```

Koja od slijedećih tvrdnji (koje se odnose na predloženu funkciju) je ispravna?


- a) Funkcija nije ispravna: pretražuje sve čvorove liste osim zadnjeg!
- b) Funkcija je ispravna i vraća pokazivač na čvor sa zadanom šifrom ili NULL ako čvor sa zadanom šifrom ne postoji u listi.
- c) Funkcija nije ispravna: ne radi za praznu listu (uzrokuje pogrešku kod poziva)!**
- d) Funkcija nije ispravna: ukoliko čvor sa zadanom šifrom ne postoji u listi, funkcija neće "nikad" završiti
- e) Funkcija nije ispravna: pretražuje sve čvorove liste osim prvog!

Koja od prikazanih stabala su gomile:

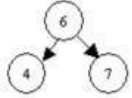
**A**



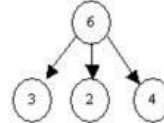
**B**



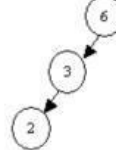
**C**



**D**



**E**



- a) B
- b) A, B**
- c) A, B, E
- d) A
- e) A, B, C, E

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)	$O(n^2 * \log_2 n)$
b)	$O(n)$
c)	$O(n^2)$
d)	$O(\log_2 n)$
<b>e)</b>	$O(n * \log_2 n)$

U prazno binarno stablo (uređeno tako da se lijevo stavlja manji a desno veći broj) su redom ubačeni sljedeći brojevi:

20, 15, 1, 3, 7, 48, 12, 19, 35

Kolika je dubina tako dobivenog stabla ?(uzmite da je korijen stabla na razini 1)

<b>a)</b>	6
b)	8
c)	5
d)	7
e)	4

Koja od sljedećih tvrdnji za jednostruko povezanu linearnu listu je istinita?

a)	Zadnji podatak u listi pokazuje na prvi podatak iz te liste
b)	Kad je lista prazna pokazivač glava pokazuje sam na sebe
c)	Podaci se mogu dodavati isključivo na početak
d)	Takva lista isključivo se realizira u datoteci na disku
<b>e)</b>	Takva lista može se realizirati statičkom strukturom podataka (poljem)

Koja od sljedećih tvrdnji je istinita?

a)	inorder obilazak stabla obrađuje dvostruko više elemenata nego postorder obilazak
<b>b)</b>	inorder i preorder obilaskom bit će obrađeni svi čvorovi u stablu
c)	postorder obilazak uvijek obrađuje samo listove stabla
d)	obilazak preorder moguće je jedino primijeniti na <i>punim</i> stablima
e)	preorder obilazak uvijek obrađuje samo listove stabla

Ako je gomila realizirana u polju, u kojem od sljedećih slučajeva elementi zapisani u polju zadovoljavaju svojstvo gomile ?

a)	10 1 9 4 5 7 8
b)	20 10 15 11 6 7 8
c)	20 7 15 8 6 10 11
d)	10 7 9 4 8 1 2
<b>e)</b>	20 3 15 2 1 14 10



Koliku dubinu i broj elemenata u zadnjoj razini ima potpuno binarno stablo sa 300 elemenata? Pretpostaviti da je korijen stabla na dubini 1.

a)	Ne može se odrediti.
b)	Dubina = 10, broj elemenata zadnje razine stabla = 10.
c)	Dubina = 9, broj elemenata zadnje razine stabla = 44.
d)	Dubina = 9, broj elemenata zadnje razine stabla = 45.
e)	Dubina = 300, broj elemenata zadnje razine stabla = 1.

Kakav je sadržaj stoga nakon izvođenja funkcije `funkcija`, ako stog prije poziva nije 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 dodaj (tip element, Stog *stog);
int skini (tip *element, Stog *stog);
```

```
#include <stdio.h>
#define MAXSTOG 100
```

```
void funkcija(Stog *stog) {
    Stog pomStog;
    int i;
    init_stog(&pomStog);
    while (skini(&i, stog)){
        if (i>=0) dodaj(i, &pomStog);
    }
    while (skini(&i, &pomStog)) {
        if (i<0){
            dodaj(i, stog);
        }
    }
}
```

a)	Stog je prazan
b)	Sadržaj stoga je nepoznat
c)	Sadržaj stoga je nepromijenjen

Dane su funkcije za rad sa stogom:

```
int dodaj (int element, Stog *stog);
int skini (int *element, Stog *stog);
```

koje vraćaju vrijednost 1 ako je operacija uspješno obavljena, tj. vrijednost 0 ako operacija nije obavljena. Što radi funkcija `propis`, uz pretpostavku da na stogovima ima dovoljno mjesta?

```
void propis(Stog *stog1, Stog *stog2) {
    int element;
    if (skini(&element, stog1)) {
        propis(stog1, stog2);
        dodaj(element, stog2);
    }
}
```

a)	Zapisuje elemente stoga 1 u stog 1 obrnutim redoslijedom
b)	Funkcija samo prazni stog 1 i ne zapisuje ništa u stog 2.
c)	Premješta sve elemente stoga 1 na vrh stoga 2. Premješteni elementi na stogu 1 poredani obrnutim redoslijedom u odnosu na stog 2.
d)	Premješta sve elemente stoga 1 na vrh stoga 2. Premješteni elementi na stogu 1 poredani istim redoslijedom kao na stogu 2.
e)	Zapisuje elemente stoga 2 u stog 2 obrnutim redoslijedom

Funkcija za dodavanje elementa na stoga realiziran listom glasi:

a) 

```
int dodaj (int element, Stog *stog) {  
    atom *novi;  
    if ((novi = (atom*) malloc(sizeof(atom))) != NULL) {  
        novi->element = element;  
        novi->sljed = stog->vrh;  
        stog->vrh = novi;  
        return 1;  
    }  
    else return 0;  
}
```

b) 

```
int dodaj (int element, Stog *stog) {  
    atom *novi;  
    if ((novi = (atom*) malloc(sizeof(atom))) != NULL) {  
        novi = element;  
        return 1;  
    }  
    else return 0;  
}
```

```
int dodaj (int element, Stog *stog) {  
    atom *novi;  
    if ((novi = (atom*) malloc(sizeof(atom))) != NULL) {  
        novi->element = stog->vrh;  
    }
```

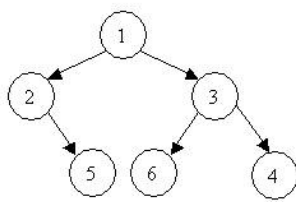
Što će se ispisati funkcijom:

```
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) 521634

b) 134625

c) 123456

d) 12364

e) 13462

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

a) broj razina=7 broj čvorova=37

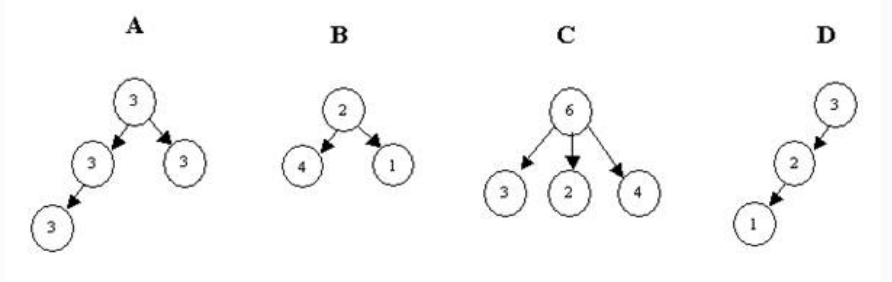
b) broj razina=6 broj čvorova=50

c) broj razina=7 broj čvorova=50

d) broj razina=6 broj čvorova=64

e) broj razina=7 broj čvorova=64

Koja od prikazanih stabala su gomile:



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

Koja od sljedećih slika prikazuje gomilu oblikovanu ulaznim nizom: 11, 33, 4, 5, 22, 8, 9, tako da složenost postupka za najgori slučaj bude  $O(n \cdot \log n)$ , gdje je  $n$  broj elemenata ulaznog niza?

a)	
b)	
c)	
d)	
e)	

Neka jednostruko povezana lista sadrži čvorove sa sljedećim tipom zapisa:

```
struct s {
    int broj;
    struct s *sljed;
};
typedef 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 posljednji čvor u listi koji sadrži broj 7, te redni broj tog čvora u listi
c)	Funkcija vraća pokazivač na prvi čvor u listi koji sadrži broj 7, te redni broj tog čvora u listi

Neka jednostruko povezana lista sadrži čvorove sa zapisima o osobama:

```
struct s {
    int sifraOsobe
    int godinaStaza;
    float placaOsobe;
    struct s *sljed;
};
typedef struct s cvor;
Zadana je funkcija f
float f(cvor *p, int g, int *br) {
    if (p) {
        if (p->godinaStaza >= g) {
            ++(*br);
            return p->placaOsobe + f(p->sljed, g, br);
        }
        else {
            return f(p->sljed, g, br);
        }
    } else {
        return 0.;
    }
}
```

Kakve vrijednosti sadrže varijable br i p nakon poziva funkcije f u sljedećem programskog odsječku:

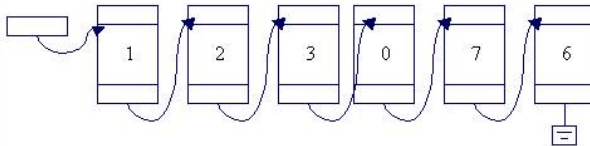
```
...
cvor *glava; int br = 0;
...
// glava pokazuje na pocetak liste u trenutku poziva funkcije
p = f(glava, 10, &br);
...
```

a)	br = broj osoba s 10 ili više godina radnog staža, p = prosjek plaća osoba s 10 ili više godina radnog staža
b)	br = broj osoba u listi, p = prosjek plaća svih osoba u listi
c)	br = broj osoba u listi, p = zbroj plaća svih osoba u listi
<input checked="" type="checkbox"/> d)	br = broj osoba s 10 ili više godina radnog staža, p = zbroj plaća osoba s 10 ili više godina radnog staža
e)	br = broj osoba u listi kojima je plaća veća ili jednaka 10, p = zbroj plaća svih osoba u listi kojima je plaća veća ili jednaka 10

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  $O(n \log n)$ ?

a)	50 5 7 10 13 1 8
b)	50 13 8 10 7 5 1
c)	50 13 7 5 10 1 8
<b>d)</b>	50 13 8 5 10 1 7
e)	50 13 10 8 7 5 1

Što će vratiti priložena funkcija za zadanu jednostruko povezanu linearnu listu:  
Slika (unutar čvorova prikazana 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)	19
<b>b)</b>	13
c)	0
d)	6
e)	16

Što će se ispisati funkcijom:

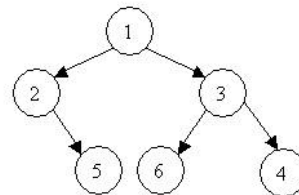
```

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

za stablo na slici pozivom funkcije

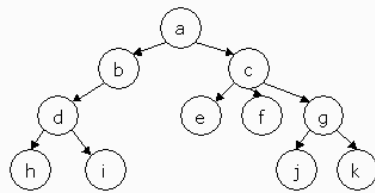
ispis (korijen);

ako je korijen u trenutku poziva pokazivač na korijen stabla?



a)	643521
<b>b)</b>	463521
c)	
d)	123456
e)	

Koja od sljedećih tvrdnji NIJE istinita?



Slika (odnosi se samo na dva ponuđena odgovora):

<input checked="" type="checkbox"/> a)	Svi čvorovi sa stabla na slici su istog stupnja
<input type="checkbox"/> b)	<b>Binarno</b> stablo koje je visine $k$ i ima $2^k-1$ elemenata naziva se <b>puno (full) binarno stablo</b>
<input type="checkbox"/> c)	U stablu sa slike listovi su: $\{h,i,e,f,j,k\}$

Postoji stog na kojem se nalaze cijeli brojevi i postoje funkcije push i pop za stavljanje, odnosno skidanje elementa sa stoga sa sljedećim prototipovima (postoji samo jedan stog tako da se funkcijama ne trebaju prenositi parametri koji definiraju stog):

```
int push( int x );  
int pop( ) ;
```

Push vraća 1 ako je uspješno stavljen element a 0 ako nije, dok pop vraća element koji je skinut sa stoga (u ovom zadatku se pretpostavlja da ima dovoljno elemenata na stogu).

Pitanje je što će se nalaziti na stogu nakon izvođenja sljedeće naredbe ?

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

a)	5 1 1
b)	5 1 2
c)	u programu je sintaksna pogreška
<input checked="" type="checkbox"/> d)	5 2
e)	5 1

Za stog realiziran cjelobrojnim poljem postoje funkcije push i pop koje stavljaju, odnosno uzimaju element sa stoga. 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++ )  
    pop( ) ;
```

<input checked="" type="checkbox"/> a)	1 2 3 4 5
b)	1 2 3 4 5 6 7 8 9 10
c)	neće biti više elemenata na stogu
d)	6 7 8 9 10
e)	1

Koja od sljedećih tvrdnji vezanih uz *stog* je istinita?

a)	za programsku realizaciju stoga moguće je isključivo koristiti stog koji je definiran kao polje
b)	interni stog računala koristi se samo pri deklaraciji globalnih varijabli C programa
<input checked="" type="checkbox"/> c)	pojedina operacija <i>dodaj</i> i <i>brisi</i> zahtijeva jednako vremena bez obzira na broj pohranjenih podataka
d)	stog je programska struktura u koju se dodaju i brišu elementi po načelu FIFO (First In First Out)
e)	na stog se mogu pohraniti isključivo cjelobrojni podaci