JEDNOSTRUKO POVEZANE LISTE

1. zadatak

U jednostruko povezanu listu spremljeni su cijeli brojevi . Napišite funkciju za dodavanje novog elementa na kraj liste.

```
/* nerekurzivno rješenje */

void dodajNaKraj(atom ** glava, tip element){
     atom * novi;
     atom * pom;
     pom = *glava;
     while(pom && pom->sljed) pom=pom->sljed;
     novi = (atom *)malloc (sizeof (atom));
     novi->sljed = NULL;
     novi->element = element ;
     //ako je lista bila prazna
     if (!(*glava)) *glava = novi;
     else
       pom->sljed = novi;

}


/* rekurzivno rješenje */

void dodajNaKrajRek(atom ** glava, tip element){
     atom * novi;

     if(*glava){
          dodajNaKraj (&(*glava)->sljed, element);
     }
     else
     {
          novi = (atom *)malloc (sizeof (atom));
          novi->sljed=NULL;
          novi->element = element ;
          *glava = novi;
     }
}
```

2.zadatak

U jednostruko povezanu **sortiranu** listu spremljeni su podaci o studentima. Čvorovi liste su sortirani uzlazno (od najmanjeg prema najvećem) po vrijednostima prosjeka Lista je zadana strukturama:

```
struct zapis {
  char imeprezime[80+1];
  float prosjek;
};
```

```
struct at {
  struct zapis element;
  struct at *sljed;
};
typedef struct at atom;
```

Napišite funkciju koja će **silazno** (od najvećeg prema najmanjem)sortirati čvorove liste po vrijednostima prosjeka **(bez stvaranja novih čvorova ili korištenjem pomoćnih polja).**

Funkcija mora imati prototip:

```
void silazniSort(atom **glava);
```

```
void silazniSort (atom **glava){
  atom *novi;
  atom *pp, *sljed=NULL;
  pp=*glava;
  if (pp){
        sljed=pp->sljed;
        while(sljed){
            novi=sljed;
            sljed=novi->sljed;
            novi->sljed =pp;
            pp=novi;
        }
    if (*glava) (*glava)->sljed=NULL;
    }
    *glava=pp;

}
```

3. zadatak

Neka su zadane dvije uzlazno sortirane liste. Napisati rekurzivnu funkciju koja će dvije zadane liste spojiti u jednu, također uzlazno sortiranu te je vratiti u pozivajući program. Prototip funkcije je:

```
atom *spoji(atom *gl1, atom *gl2);
```

```
atom *spoji(atom *gl1, atom *gl2) {
  atom *gl = NULL;

  if (gl1 == NULL) return gl2;
  else if (gl2 == NULL) return gl1;

  if (gl1->element <= gl2->element) {
      gl = gl1;
      gl->sljed = spoji(gl1->sljed, gl2);
  }
  else {
      gl = gl2;
      gl->sljed = spoji(gl1, gl2->sljed);
  }
  return gl;
}
```

DVOSTRUKO POVEZANE LISTE

1.zadatak
U dvostruko povezanoj listi **sortiranoj silazno** (od glave) po prosjeku spremljeni su podaci o studentima. Lista je zadana strukturama:

```
struct zapis {
  char imeprezime[80+1];
  float prosjek;
};
```

```
struct at {
  struct zapis element;
  struct at *sljed;
  struct at *preth;

};
typedef struct at atom;
```

Napišite **funkciju** koja će obrnuti redoslijed elemenata u dvostruko povezanoj listi.
Prototip funkcije je:

```
void okreniListu(atom **glava, atom **rep)
```

```
void okreniListu(atom **glava, atom **rep)
{
     atom *pom,*pom1;
     pom = *glava;
     *glava = *rep;
     *rep = pom;
     while(pom != NULL)
     {
          pom1 = pom->sljed;
          pom->sljed = pom->preth; pom->preth = pom1;
          pom=pom->preth;
     }
}
```

2.zadatak

U čvorovima sortirane dvostruko povezane liste nalaze se zapisi o knjigama: **šifra knjige** (int), **naziv knjige** (100+1 znak), **šifra autora** (int) i **cijena knjige** (float). Lista je uzlazno sortirana prema cijeni knjige.
Potrebno je napisati funkciju koja će iz liste izbaciti knjige s najvišom i najmanjom cijenom
Napomena: više knjiga može imati istu cijenu.

```c
typedef struct s {
   int sifraKnjige;
   char nazivKnjige[100+1];
   int sifraAutora;
   float cijena;
   struct s *sljed;
   struct s *preth;
} cvor;

void izbaci(cvor **glava, cvor **rep) {
   cvor *p = *glava;
   float minCijena, maxCijena;

   if (!p) { // ako je lista prazna
        return;
   }

   // izbaciti one s najmanjom cijenom
   minCijena = p->cijena;
   while (p && p->cijena == minCijena) {
        *glava = p->sljed;
        if (p->sljed) {
             p->sljed->preth = NULL;
        }
        free(p);
        p = *glava;
   }

   // ako je ispraznjena cijela lista
   if (*glava == NULL) {
        *rep = NULL;
   }
   // ako je jos ostalo elemenata u listi
   else {
        p = *rep;
        maxCijena = p->cijena;
        // izbaciti one s najvisom cijenom
        while (p && p->cijena == maxCijena) {
             *rep = p->preth;
             if (p->preth) {
                  p->preth->sljed = NULL;
             }
             free(p);
             p = *rep;
        }
        // ako je ispraznjena cijela lista
        if (*rep == NULL) {
             *glava = NULL;
        }
   }
}
```

LISTE S VIŠE KLJUČEVA

1.zadatak
U čvorovima jednostruko povezane liste uzlazno sortirane po dva ključa (matični broj i prezime) liste nalaze se zapisi o studentima: **matični broj** (int), **prezime** (100+1 znak).

Potrebno je napisati funkciju koja kao argument prima matični broj studenta i iz liste briše čvor koji sadrži podatke o studentu s tim matičnim brojem. Funkcija vraća 0 ako čvor sa matičnim brojem nije pronađen a 1 ako je pronađen i obrisan.

```
struct tip{
     int mbr;
     char prezime[14+1];
};


int obrisiPoMbr (atom **glavambr, atom ** glavaprez, int mbr){
     atom *p;

     for (; *glavambr && (*glavambr)->element.mbr != mbr;
glavambr = &((*glavambr)->smbr));

     if (*glavambr) {
          for (; *glavaprez && (*glavaprez)->element.mbr != mbr;
                    glavaprez = &((*glavaprez)->sprez));
          *glavambr = (*glavambr)->smbr;
          p = *glavaprez;
          *glavaprez = (*glavaprez)->sprez;
          free (p);

          return 1;
     } else {
          return 0;
     }
}
```

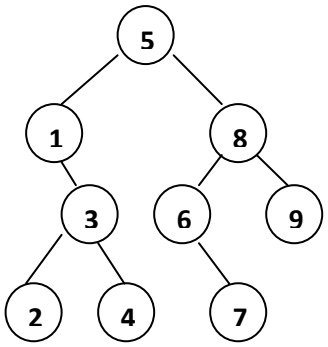
STABLA

1.Zadatak

U sortirano binarno stablo redom se stavljaju brojevi:
5 1 8 6 7 3 4 2 9

a) nacrtajte stablo
b) inorder ispis elemenata stabla (lijevo, korijen, desno)
c) inorder ispis elemenata stabla (desno, korijen, lijevo)
d) preorder ispis elemenata stabla (korijen, lijevo, desno)
e) postorder ispis elemenata stabla (lijevo, desno korijen)

a)



b) 1 2 3 4 5 6 7 8 9
c) 9 8 7 6 5 4 3 2 1
d) 5 1 3 2 4 8 6 7 9
e) 2 4 3 1 7 6 8 9 5

2. zadatak

Binarno stablo karakterizirano je sljedećim podacima:

- ispis *inorder*: c, b, y, j, g, m, x, o, z
- ispis *postorder*: c, y, j, b, x, z, o, m, g

Potrebno je:

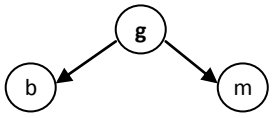
a) rekonstruirati binarno stablo (nacrtati izgled)

b) na liniju napisati preorder ispis stabla

Osnovna ideja: *postorder* obilazak stavlja korijen na kraj, dok kod *inorder* obilaska korijen razdvaja lijevo i desno podstablo.

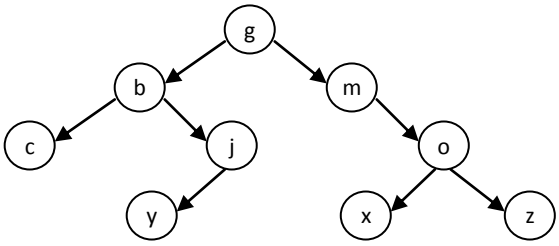
Korijen je **g**

Lijevo podstablo	Desno podstablo
inorder: c, b, y, j	*inorder*: m, x, o, z
postorder: c, y, j, b	*postorder*: x, z, o, m



Lijevo (LLPS)	Desno	Lijevo	Desno
inorder: c	*inorder:* y, j		*inorder*: x, o, z
postorder: c	*postorder*: y, j		*postorder*: x, z, o

Analognim postupkom rekonstruiramo stablo do kraja



b) `Preorder poredak: g, b, c, j, y, m, o, x, z.`

3. zadatak
U binarno stablo spremaju se cjelobrojni podaci (**long**) (nema duplih vrijednosti). Stablo je sortirano (lijevo manji, desno veći)
a) Napisati **NEREKURZIVNU** funkciju koja će pronaći zadani element u stablu. Ako se zadani element nalazi u stablu, funkcija treba vratiti pokazivač na njega. Ako zadani element ne postoji u stablu, funkcija treba vratiti NULL. Funkcija treba imati prototip:

b) Napisati REKURZIVNU funkciju koja će vratiti broj elemenata binarnog stabla čija je vrijednost jednaka njihovoj razini. Razina korijena stabla je 1. Funkcija treba imati prototip:

```
int brNaRazini(cvor *glava, long razina);
```

```
cvor * trazi (cvor *korijen, long element) {

    while (korijen && (korijen ->element != element)) {
        if (element > korijen ->element) korijen = korijen ->desno;
        else korijen = korijen ->lijevo;
    }

    return korijen;
}


int brNaRazini(cvor *korijen, int tren_raz){
   if (korijen){
      return (korijen ->element == tren_raz)
             + broji(korijen ->lijevo, tren_raz+1)
             + broji(korijen ->desno,  tren_raz+1);
   }
   return 0;
}
```

/* ovako bi izgledalo rekurzivno rješenje od a) zadatka */

```
cvor *trazi (cvor *korijen, long element) {
     if (korijen) {
          if (element< korijen->element)
               return trazi (korijen->lijevo, element);
          else
               return trazi (korijen->desno, element);
     }
     return korijen; // ili je pronadjen ili NULL;
}
```

4. zadatak
Napišite funkciju koja ispisuje vrijednosti u listovima stabla ako je struktura čvora:

```
struct cv {
  char element[15];
  struct cv *lijevo;
  struct cv *desno;
};
```

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