UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

STRUKTURE PODATAKA I ALGORITMI

– ZBIRKA ZADATAKA –

2011.

SADRŽAJ

NIZ

Ubacivanje elementa u rastuce sortiran niz

```
public void InsertSortNiz(int value)
{
    for(int j=0; j<nElems; j++) {
        if(a[j] > value)
        break;
    }
    for(int k=a.length; k>j; k--)
        a[k] = a[k-1];
    a[j] = value;
}
```

Spajanje dva rastuce sortirana niza u treci

```
public int[] spojNizove(int[] a, int[] b) {
int i=j=br=0;
int k=a.length+b.length;
int niz[] = new int[k];
while(br<k){
  while(a[i]<=b[j])
        niz[br++]=a[i++];
  niz[br++]=b[j++];
}
return niz;
}
```

Dat je niz celih brojeva. Napisati metodu **int prosekParnih(int [] niz)** koja ce naci prosek parnih elemenata niza (onih koji imaju parne vrednosti, a ne na parnim mestima u nizu).

```
public int prosekParnih(int[] niz) {
    int zbir = 0;
    int broj = 0;
    for(int i=0; i<niz.length; i++) {
        if(niz[i] mod 2 == 0) {
            zbir+=niz[i];
            broj++;
        }
    }
    return zbir/broj;
}
```

PRETRAŽIVANJE

Sekvencijalno – iterativno

```
public int sekvencijalno(int pod, int[] niz) {
    for(int i=0; i<niz.length; i++) {
        if(niz[i]==pod)
            return i;
    }
    return -1;
}
```

Sekvencijalno – rekurzivno

```
public int sekvencijalnoR(int pod, int[] niz, int i=0) {
    if(i>-1 && i<niz.length) {
          if(niz[i]==pod)
                return i;
          return sekvencijalnoR(pod, niz, ++i);
    }
    return -1;
}
```

Sekvencijalno – iterativno (rastuci)

```
public int sekvencijalno(int pod, int[] niz) {
    int i=0;
    while(i<niz.length) {
        if(niz[i]==pod) return i;
        if(niz[i]>pod) break;
        i++;
    }
    return -1;
}
```

Binarno – iterativno (rastuci)

```
public int binarno(int pod, int[] niz) {
    int l = 0; int d = niz.length-1;
    while(l<=d) {
        int s = (l+d)/2;
        if(niz[s]==pod) return s;
        if(niz[s]>pod)
            d=s-1;
        else
            l=s+1;
    }
    return -1;
}
```

Binarno – rekurzivno (rastuci)

```
public int binarnoR(int pod, int[] niz, int l, int d) {
    if(l<=d) {
        int s=(l+d)/2;
        if(niz[s]==pod) return s;
        if(niz[s]>pod)
            return binarnoR(pod, niz, l, s-1)
        else
            return binarnoR(pod, niz, s+1, d)
    }
    return -1;
}
```

Interpolaciono – iterativno (rastuci)

```
public int interpolaciono(int pod, int[] niz) {
    int l = 0; int d = niz.length-1;
    while(l<=d) {
        int index = l + (d-l)/(niz[d]-niz[l])*(pod-niz[l]);     //  (index-l)/(pod-niz[l]) = (d-l)/(niz[d]-niz[l])
        if(niz[index]==pod) return s;
        if(niz[index]>pod)
            d=index-1;
        else
            l=index+1;
    }
    return -1;
}
```

Interpolaciono – rekurzivno (rastuci)

```
public int interpolacionoR(int pod, int[] niz, int l, int d) {
    if(l<=d) {
        int index = l + (d-l)/(niz[d]-niz[l])*(podatak-niz[l]);
        if(niz[index]==pod) return index;
        if(niz[index]>pod)
            return interpolacionoR(pod, niz, l, index-1);
        else
            return interpolacionoR(pod, niz, index+1, d);
    }
    return -1;
}
```

SORTIRANJE

Selection sort

```
public void SelectionSort(int[] niz)
{
   for (int i = 0; i < niz.length - 1; i++)
   {
      int min = i;
      for (int j = i + 1; j < niz.length; j++)
      {
         if (niz[j] < niz[min])
            min = j;
      }
      if(min!=i) {
          int pom = niz[i];
          niz[i] = niz[min];
          niz[min] = pom;
      }
   }
}
```

Bubble sort

```
public void BubbleSort(int[] niz)
{
    for(int i=niz.length-1; i>0; i--) {
        for(int j=0; j<i; j++) {
            if(niz[j]>niz[j+1]) {
                int pom = niz[j];
                niz[j] = niz[j+1];
                niz[j+1] = pom;
            }
        }
    }
}
```

Insertion sort

```
public void InsertionSort(int[] niz)
{
  for (int i = 1; i < niz.length; i++)
  {
    int pom = niz[i];
    int j = i;
    while ((j > 0) && (niz[j-1] > pom))
    {
      niz[j] = niz[j-1];
      j--;
    }
    niz[j] = pom;
  }
}
```

Shell sort

```
public void ShellSort(int[] niz)
{
    int inc = niz.length/2;
    while(inc > 0) {
        for (int i = 1; i < niz.length; i++)
        {
            int pom = niz[i];
            int j = i;
            while ((j >=inc) && (niz[j-inc] > pom))
            {
                niz[j] = niz[j-inc];
                j-=inc;
            }
            niz[j] = pom;
        }
        inc = inc/2;
    }
}
```

Quick Sort

```
public void QuickSort(int[] niz, int low, int high) {
    if(high<=low) return;
    int pivot = partition(niz, low, high);
    QuickSort(niz, low, pivot-1);
    QuickSort(niz, pivot+1, high);
}

public int partition(int[] niz, int low, int high) {
    int i = low – 1;
    int j = high;
    int pivot = a[high];
    for(;;) {
        while(niz[++i]<pivot);
        while(niz[--j]>pivot) if (j==low) break;
        if(i>=j) break;
        zameni(niz, i, j);
    }
    zameni(niz, i, r);
    return(i);
}

public void zameni(int[] niz, int i, int j) {
    int pom = niz[i];
    niz[i] = niz[j];
    niz[j] = pom;
}
```

RED

Implementacija reda - staticka

```
public class StatRed{
    int[] niz;
    int dimenzija;
    int p;
    int k;
    int brojElem;

    public StatRed(int kapacitet) {
        dimenzija = kapacitet;
        niz = new int[dimenzija];
        p = 0;
        k = -1;
        brojElem = 0;
    }

    public boolean prazan() {
        return brojElem == 0;
    }

    public boolean pun() {
        return brojElem == dimenzija;
    }

    public int pocetak() {
        if(prazan) throw new Exception(“Prazan“);
        return niz[p];
    }

    public int kraj() {
        if(prazan) throw new Exception(“Prazan“);
        return niz[k];
    }

    public boolean enque(int pod) {
        if(pun) return false;
        k++;
        if (k==dimenzija) k=0;
        niz[k]==pod;
        brojElem++;
        return true;
    }

    public int deque() {
        if(prazan) throw new Exception("Prazan");
        int pod = niz[p];
        p++;
        if(p==dimenzija) p=0;
        brojElem--;
        return pod;
    }
```

```
public void ispisiRed() {
    for(int i=p; i!=k; i++) {
       if (i==niz.length) i==0;
       System.out.println(niz[i]);
    }
    System.out.println(k);
}
}
```

Implementacija reda - dinamicka

```
public class Red implements Queue {
  private CvorJS prvi, poslednji;
  public Red( ) { isprazni( ); }

  public void isprazni( )
  { prvi = poslednji = null; }

  public boolean prazan( )
  { return prvi == null; }

  public void enqueue(Object x) {
    if ( isEmpty() )
      poslednji = prvi = new Cvor(x, null);
    else
      poslednji = poslednji.next = new Cvor(x, null);
  }

  public Object dequeue( ) throws Underflow {
    if (prazan() )
      throw new Underflow("Prazan");
    Object item = prvi.element;
    prvi = prvi.next;
    return item;
  }

  public Object getPrvi( ) throws Underflow {
    if ( isEmpty() )
      throw new Underflow("Empty queue");
    return prvi.element;
  }
}
```

Invertovanje reda tako da ostane nepromenjen

Dat je red celih brojeva r1. Napisati f-ju Inv(Red r1, Red r2) koja ce napraviti novi red r2 koj je inverzan datom redu.Po zavrsetku algoritma pocetni red treba da ostane nepromenjen. Nije dozvoljeno koriscenje nikakvih pomocnih struktura! Moguce je koriscenje samo sledecih operacija nad redom : Ubaci(int), Izbaci() i boolean Prazan().

```
public void invertujRed (Red r1, Red r2) {
   if (r1.prazan()!=true) {
      int izbacen = r1.izbaci();
      r2.ubaci(izbacen);
      invertujRed(r1,r2);
   }
   r1.ubaci(r2.izbaci());
   r2.ubaci(izbacen);
}
```

Dat je red impl. kao JS lista. Broj elemenata koji zadovoljavaju zadati kriterijum.

Dat je pokazivač na početak reda celih brojeva, koji je implementiran kao jednostruko spregnuta lista. Napisati funkciju int Prebroj(CvorListe Vrh) koja će vratiti koliko elemenata reda ima vrednost veću od početnog elementa.

```
public int Prebroj(CvorListe Vrh) {
   int pod = Vrh.pod;
   int broj = 0;
   CvorJS pom = Vrh;
   while (pom!=null) {
      if(pom.pod>pod) broj++;
      pom=pom.sledeci;
   }
   return broj;
}
```

STEK

Implementacija steka - staticka

```
public class StatStek{
    int[] niz;
    int vrh;
    int dimenzija;

    public StatStek(int kapacitet) {
        dimenzija=kapacitet;
        niz = new int[dimenzija];
        vrh=-1;
    }

    public int brElem() {
        return vrh+1;
    }

    public boolean prazan() {
        return (vrh == -1);
    }

    public boolean pun() {
        return (vrh == dimenzija-1);
    }

    public int peek() {
        if(prazan) throw new Exception("Prazan");
        return niz[vrh];
    }

    public boolean push(int elem) {
        if (pun) return false;
        niz[++vrh] = elem;
        return true;
    }

    public int pop() {
        if (prazan) throw new Exception("Prazan");
        return niz[vrh--];
    }
}
```

Implementacija steka – dinamicka

Dinamicki stek – brojanje elemenata

Dat je pokazivač na vrh steka celih brojeva, koji je implementiran kao JS lista. Napisati funkciju int Prebroj(CvorListe Vrh) koja će vratiti koliko elemenata steka ima vrednost veću/manju od elementa na vrhu staka.

```
public int Prebroj (CvorListe Vrh) {
  int total = 0;
  int vrednost = Vrh.vrednost;

  while(Vrh != null) {     // prebroj vece vrednosti
    if(Vrh.vrednost > vrednost) {
      total++;
    }
    Vrh = Vrh.sledeci;
  }
  return total;
}
```

Dinamicki stek – kloniranje

```
public static void clone(izvor, klon) {
    Object temp = izvor.Pop();
    if (temp == null)
        return;
    clone(izvor, klon);
    klon.Push(temp);
    izvor.Push(temp);
}
```

Dinamicki stek – "Izbaci" nad stekom celih brojeva impl. kao JS lista

```
public int Pop() {
  if (prvi == null) return null;
  int pod = prvi.pod;
  if (prvi.sledeci == null) {
      prvi = null;
  } else {
      prvi = prvi.sledeci;
  }
  return pod;
}
```

Dinamicki stek – "Izbaci" nad stekom celih brojeva impl. kao DS lista

```
public int Pop() {
  if (prvi == null) return null;
```

```
  int pod = prvi.pod;
  if (prvi.sledeci == null) {
      prvi = poslednji = null;
  } else {
      prvi = prvi.sledeci;
      prvi.prethodni = null;
  }
  return pod;
}
```

Dinamicki stek – "Gurni" nad stekom celih brojeva impl. kao JS lista

```
public void Push(int pod) {
  CvorJS novi = new CvorJS(pod, prvi);
  prvi = novi;
}
```

Dinamicki stek – transformacija u staticki

Napisati funkciju transformisi(STAK s1, STAK* s2) koja će od steka koji je implementiran kao JS lista formirati novi stek koji je implementiran preko niza.

```
public ArrayStack transform(LinkedListStack s1) {
    LinkedListStack temp = new LinkedListStack();
    int count = 0;
    Object element;
    while ((element = s1.Pop()) != null) { //puni u obrnutom redosledu
        temp.Push(element);
        count++; //izbroj elemente steka
    }
    ArrayStack stek = new ArrayStack(count); //kreiraj niz
    while ((element = temp.Pop()) != null) { //puni u ispravnom redosledu
        stek.Push(element);
    }
    return stek;
 }
```

JEDNOSTRUKO SPREGNUTA LISTA

Implementacija JS

```
public class CvorJS {
    int pod;
    CvorJS sledeci;

    public CvorJS(int p, CvorJS sled) {
        pod = p;
        sledeci = sled;
    }
}

public class JSLista {
    CvorJS prvi;
    public JSLista() {
        prvi = null;
    }

    public boolean prazna() {
        return prvi == null;
    }

    public boolean puna() {
       return false;
    }

    public void ubaciNaPocetak(int pod) {
        CvorJS novi = new CvorJS(pod, prvi);
        prvi = novi;
    }

    public void ubaciNaKraj(int pod) {
        CvorJS novi = new CvorJS(pod, null);
        if(prazna()) {
            ubaciNaPocetak(pod);
        } else {
            CvorJS pom = prvi;
            while(pom.sledeci!=null)
                pom = pom.sledeci;
        }
        pom.sledeci = novi;
    }

    public void ubaciPre(int podatak, CvorJS t) {
        CvorJS novi = new CvorJS(podatak, t);
        if(prvi==t)
           prvi = novi;
        else {
           CvorJS pret = prvi;
           while(pret.sledeci != t)
               pret = pret.sledeci;
           pret.sledeci = novi;
```

```
    }
}

public void ubaciNakon(int podatak, CvorJS t) {
    CvorJS novi = new CvorJS(podatak, t.sledeci);
    t.sledeci = novi;
}

public void ubaciUSort(int podatak) {
    if (prazna() || prvi.pod > podatak) {
        CvorJS prvi = new CvorJS(pod, prvi);
    } else {
        CvorJS pom = prvi;
        while(pom.sledeci != null && pom.sledeci.pod < pod)
            pom = pom.sledeci;
    }
    CvorJS novi = new CvorJS(pod, pom.sledeci);
    pom.sledeci = novi;
}

public int izbaciSaPocetka() {
    if(prazna()) return Integer.MIN_VALUE;
    int pod = prvi.pod;
    prvi = prvi.sledeci;
    return pod;
}

public int izbaciSaKraja() {
    if (prazna() || prvi.sledeci == null)
        return izbaciSaPocetka();
    CvorJS pom = prvi;
    while(pom.sledeci.sledeci != null) {
        pom = pom.sledeci;
    }
    int pod = pom.sledeci.pod;
    pom.sledeci = null;
    return pod;
}

 public int izbaciPre(CvorJS t) {
     if(prazna() || t==prvi) return Integer.MIN_VALUE;
     CvorJS pom = prvi;
     if(pom.sledeci==t) return izbaciSaPocetka;
     while(pom.sledeci.sledeci != t)
         pom = pom.sledeci;
     int pod = pom.sledeci.pod;
     pom.sledeci = t;
     return pod;
 }

 public int izbaciNakon(CvorJS t) {
     if(prazna() || t.sledeci == null) return Integer.MIN_VALUE;
     int pod = t.sledeci.pod;
     t.sledeci = t.sledeci.sledeci;
```

```
        return pod;
    }

    public void ispisi() {
        CvorJS pom = prvi;
        while (pom!=null) {
            System.out.println(pom.pod);
            pom = pom.sledeci;
        }
    }

    public void ispisiObrnuto(CvorJS pom) {
        if(pom==null) return;
        if (pom.sledeci!=null) ispisiObrnuto(pom.sledeci);
        System.out.println(pom.pod);
    }
}
```

Broj ponavljanja elementa

```
public class Cvor {
    int pod;
    Cvor sledeci;
}

public int brojPonavljanja(Cvor c, int pod) {
    if (c==null) return 0;
    if (c.pod == pod) return 1 + brojPonavljanja(c.sledeci, pod);
    else return brojPonavljanja (c.sledeci, pod);
}
```

Invertovanje JS liste

```
public class Cvor {
    int pod;
    Cvor sledeci;
}

public void invertujJS(CvorJS c) {
    CvorJS pom, sled;
    pom = null;
    sled = c.sledeci;
    while (sled!=null) {
        c.sledeci = pom;
        pom = c;
        c = sled;
        sled = c.sledeci;
    }
}
```

Presek dve JS liste

```
public class CvorJS {
    int pod;
    CvorJS sledeci;
}
```

```
public boolean postoji(int pod, CvorJS c) {
   while (c != null) {
      if (c.podatak == pod) return true;
      c= c.sledeci;
   }
   return false;
}

public CvorJS presek (CvorJS c1, CvorJS c2) {
    if (c1==null) || (c2==null) return null;
    CvorJS c3 = null;
    if (!postoji(c1.pod, c2))
       return presek(c1.sledeci, c2);
    else {
       c3 = new CvorJS(c1.pod, c3);
       return presek (c1.sledeci, c2);
   }
}
```

Unija dve JS liste

```
public class CvorJS {
    int pod;
    CvorJS sledeci;
}

public boolean postoji(int pod, CvorJS c) {
   while (c != null) {
      if (c.podatak == pod) return true;
      c= c.sledeci;
   }
   return false;
}

public CvorJS unija(CvorJS c1, CvorJS c2) {
    if (c1==null) return c2;
    if (c2==null) return c1;
    CvorJS c3 = null;
    if (postoji(c1.pod, c2))
       return unija(c1.sledeci, c2);
    else {
       c3 = new CvorJS(c1.pod, c3);
       return unija(c1.sledeci, c2);
    }
}
```

Kloniraj JS listu

```
public class CvorJS {
    int pod;
    CvorJS sledeci;

    public CvorJS (int podatak, CvorJS sled) {
        int pod = podatak;
        sledeci = sled;
```

```
    }
}

public CvorJS kloniraj(CvorJS pom) {
    if(pom == null) return;
    int pod = pom.pod;
    kloniraj(pom.sledeci);
    CvorJS novi = new CvorJS (pod, pom);
    return novi;
}
```

Transformisi JS u staticki stek

```
public class CvorJS {
    int pod;
    CvorJS sledeci;
}

public int[] transformJSuSStek(CvorJS prvi, int[] stek) {
     CvorJS pom = prvi;
     int brojac = 0;
     while(pom!=null) {
        brojac++;
        pom=pom.sledeci;
     }
     stek = new int[brojac];
     pom = prvi;
     for(int i=0; i<brojac; i++) {
          stek[i] = pom.pod;
          pom=pom.sledeci;
     }
     return stek;
}
```

Svaki elem JS sadrzi pok na drugu JS – izbroj elem. iterativno

```
class CvorJS {
    CvorJS sledeci;
    CvorJS prviJS;
}

public int brojClanova(CvorJS prvi) {
    int broj_glavne = 0;
    int broj_sporednih = 0;
    CvorJS pom = prvi;
    while(pom!=null) {
        broj_glavne++;
        CvorJS pomsp=pom.prviJS;
        while(pomsp!=null) {
            broj_sporednih++;
            pomsp=pomsp.sledeci;
        }
        pom=pom.sledeci;
    }
    return broj_glavne + broj_sporednih;
```

```
}
```

Svaki elem JS sadrzi pok na drugu JS – izbroj elem. rekurzivno

```
public class CvorJS {
    CvorJS Next;
    CvorJS Down;
}

public int Count(CvorJS c){
    if(c==null) return 0;
    return 1 + Count(c.Next) + Count(c.Down);
}
```

Broj pozitivnih elemenata – rekurzivno

```
public int izbrojPozitivne(CvorJS pom){
    if(pom==null)
       return 0;
    else {
       if(pom.podatak>=0)
         return 1+izbrojPozitivne(pom.sledeci);
       else
         return izbojPozitivne(pom.sledeci);
    }
}
```

Izbaci element nakon najmanjeg elementa ako takav postoji

```
public static void izbaciElement(CvorJS prvi) {
     CvorJS min = prvi;
     CvorJS pom = prvi.sledeci;
     while (pom != null) {
         if(pom.podatak < min.podatak)
             min = pom;
         pom = pom.sledeci;
     }
     if(min.sledeci != null) {
         min.sledeci = min.sledeci.sledeci;
     }
}
```

Rekurzivno sabrati sve elemente JS liste

```
public int zbir (CvorJS pom)
{
    if(pom==null)
        return 0;
    else
        return pom.podatak+zbir(pom.sledeci);
}
```

DVOSTRUKO SPREGNUTA LISTA

Implementacija

```
public class CvorDS {
    int pod;
    CvorDS prethodni;
    CvorDS sledeci;

    public CvorDS(int podatak, CvorDS pret, CvorDS sled) {
        pod = podatak;
        prethodni = pret;
        sledeci = sled;
    }
    public CvorDS(int podatak, null, null) {
        this(podatak, null, null);
    }
}

public class DSLista {
    public CvorDS prvi;
    public CvorDS poslednji;

    public DSLista() {
        prvi = poslednji = null;
    }

    public boolean prazna() {
        return prvi == null;
    }

    public boolean puna() {
        return false;
    }

   public void ubaciNaPocetak(int pod) {
        CvorDS novi = new CvorDS(pod, null, prvi);
        if (prazna())
            prvi=poslednji=novi;
        else {
            prvi.prethodni = novi;
            prvi = novi;
        }
    }

    public void ubaciNaKraj(int pod) {
        CvorDS novi = new CvorDS(pod, poslednji, null);
        if(prazna())
            prvi = poslednji = novi;
        else {
            poslednji.sledeci = novi;
            poslednji = novi;
        }
    }
```

```
public void ubaciPre(int pod, CvorDS t) {
     if(prazna()) return;
     CvorDS novi = new CvorDS(pod, t.prethodni, t);
     t.prethodni = novi;
     if(prvi==t)
        prvi=novi;
     else
        novi.prethodni.sledeci = novi;
}

public void ubaciNakon(int pod, CvorDS t) {
     if(prazna()) return;
     CvorDS novi = new CvorDS(pod, t, t.sledeci);
     t.sledeci = novi;
     if(t==poslednji)
        poslednji = novi;
     else
        novi.sledeci.prethodni=novi;
}

public void ubaciUSort(int podatak) {
    if (prazna() || poslednji.pod < podatak)
        ubaciNaKraj(podatak);
    else {
        CvorDS pom = prvi;
        while(pom.podatak < pod) {
            pom = pom.sledeci;
        }
        CvorDS novi = new CvorDS(podatak, pom.prethodni, pom);
        pom.prethodni = novi;
        if(novi.pretnodni == null) {
            prvi = novi;
        else
            novi.prethodni.sledeci = novi;
        }
    }
}

public int izbaciSaPocetka() {
    if(prazna()) throw new Exception("Prazna");
    int pod = prvi.pod;
    prvi = prvi.sledeci;
    if (prvi == null)
        poslednji = null;
    else
        prvi.prethodni = null;
    return pod;
}

public int izbaciSaKraja() {
    if(prazna()) throw new Exception("Prazna");
    int pod = poslednji.pod;
    poslednji = poslednji.prethodni;
    if(poslednji == null) {
```

```
            prvi = null;
        } else {
            poslednji.sledeci = null;
        }
        return pod;
    }

    public int izbaciPre (CvorDS t) {
        if(prazna() || t==prvi) return Integer.MIN_VALUE;
        int pod = t.prethodni.pod;
        t.prethodni = t.prethodni.prethodni;
        if(t.prethodni == null)
            prvi = t;
        else
            t.prethodni.sledeci = t;
        return pod;
  }

  public int izbaciNakon (CvorDS t) {
        if(prazna() || t==poslednji) return Integer.MIN_VALUE;
        int pod = t.sledeci.pod;
        t.sledeci = t.sledeci.sledeci;
        if(t.sledeci == null)
            poslednji=t;
        else
            t.sledeci.prethodni=t;
        return pod;
    }

    public int izbaci (CvorDS t) {
    if(prazna()) return Integer.MIN_VALUE;
    if(t == prvi)
        prvi = prvi.sledeci;
    else
        t.prethodni.sledeci = t.sledeci;
    if (t == poslednji)
        poslednji = poslednji.prethodni;
    else
        t.sledeci.prethodni = t.prethodni;
     return pod;
    }
}
```

Popuni elemente rastuce DS liste tako da sadrze sukcesivne cele brojeve

Dat je pokazivac na pocetni cvor DS liste sortirane u rastucem redosledu koja sadrži pozitivne cele brojeve. Napisati funkciju koja ce izmedju svih onih elementa liste koji se po vrednosti razlikuju za više od 1 ubaciti u datu listu nove elemente tako da lista posle poziva operacije ima u sebi sukcesivne cele brojeve. Na primer ako lista sadrži 3, 5, 8 nakon poziva ove opercije sadržace 3, 4, 5, 6, 7, 8

```
public static void fill(CvorDS c) {
    if (c.Next == null)
        return;
    if (c.Next.Data - c.Data > 1) {
```

```
        /* Ako se tekuci i sledeci cvor razlikuju za vise od 1, ubaci jedan cvor izmedju i pozovi rekurzivno
        funkciju nad ubacenim cvorom */
        CvorDS temp = new CvorDS (c.Data + 1);
        // nema provere za null, jer se uvek ubacuje izmedju dva
        temp.Next = c.Next;
        temp.Previous = c;
        c.Next.Previous = temp;
        c.Next = temp;
    }
    fill(c.Next);
}
```

Popuni elemente opadajuce DS liste tako da sadrze sukcesivne cele brojeve

Dat je pokazivac na pocetni cvor dvostruko spregnute liste sortirane u opadajucem redosledu koja sadrži pozitivne cele brojeve. Napisati funkciju koja ce između svih onih elementa liste koji se po vrednosti razlikuju za više od 1 ubaciti u datu listu nove elemente tako da lista posle poziva operacije ima u sebi sukcesivne cele brojeve. Na primer ako lista sadrži 10, 7, 5 nakon poziva ove opercije sadržace 10, 9, 8, 7, 6, 5

```
public static void fillAgain(CvorDS c) {
    if (c.Next == null)
        return;
    if (c.Data - c.Next.Data > 1) {
        /*Ako se tekuci i sledeci cvor razlikuju za vise od 1, ubaci jedan cvor izmedju i pozovi rekurzivno
        funkciju nad ubacenim cvorom */
        CvorDS temp = new
        CvorDS (c.Data - 1);
        // nema provere za null, jer se uvek ubacuje izmedju dva
        temp.Next = c.Next;
        temp.Previous = c;
        c.Next.Previous = temp;
        c.Next = temp;
    }
    fillAgain(c.Next);
}
```

Vrati broj elemenata DS liste. Dat je pokazivac na neki cvor

Dat je pokazivac na neki cvor dvostruko spregnute liste. Napisati funkciju koja ce vratiti broj elemenata u listi.

```
public static int count(CvorDS n) {
    if (n == null)
        return 0;
    return 1 + before(n.Previous) + after(n.Next);
}
/*broji koliko ih je levo od datog cvora*/
private static int before(CvorDS n) {
    if (n == null)
        return 0;
    return 1 + before(n.Previous);
}
/*broji koliko ih je desno od datog cvora */
private static int after(CvorDS n) {
    if (n == null)
        return 0;
```

```
    return 1 + after(n.Next);
}
```

Izbaci element DS liste koji sadrzi najveci/najmanji broj

Dat je pokazivac na neki element DS liste celih brojeva. Napisati funkciju koja ce izbaciti iz liste onaj element koji sadrži najmanji/najveci broj u listi.

```
public static int min(CvorDS cvor) {
    if (cvor == null)
        return Integer.MIN_VALUE;
        /*prvo se pronadje najmanji element i on se snimi u promenjivu min */
        CvorDS min = findMin(cvor);
        /*onda se min izbacuje tako sto se prethodnom stavi pokazivac na sledeci posle min-a, pod uslovom
        da prethodni nije null (ovde nema veze da li je sledeci null) ,a onda se sledecem stavi pokazivac na
        prethodni pre min, pod uslovom da sledeci nije null(sada nema veze da li je prethodni null)*/
        if (min.Previous != null)
            min.Previous.Next = min.Next;
        if (min.Next != null)
            min.Next.Previous = min.Previous;
        return min.Data;
}

private static CvorDS findMin(CvorDS cvor) {
    // postavlja tekuci element za minimum
    CvorDS min = cvor;
    int minValue = cvor.Data;
    // ide do levog kraja i trazi manji
    CvorDS iterator = cvor;
    while (iterator.Next != null) {
        iterator = iterator.Next;
        if (iterator.Data < minValue) {
            min = iterator;
            minValue = iterator.Data;
        }
    }
    iterator = cvor;
    // ide do desnog kraja i trazi jos manji
    while (iterator.Previous != null) {
    iterator = iterator.Previous;
    if (iterator.Data < minValue) {
        min = iterator;
        minValue = iterator.Data;
    }
    }
    return min;
}
```

Ubaci element na n-to mesto DS liste

Dat je pokazivac na neki cvor DS liste celih brojeva koja sigurno sadrži najmanje 4 elementa. Napisati funkciju ubaciNti(int A, int N) koja ce ubaciti novi element sa sadržajem A i to tako da on bude na N-toj poziciji od pocetka.

```
public static boolean ubaciNti(CvorDS cvor, int a, int n) {
    if (cvor == null)
        return false;
```

```
    // resetuje listu na pocetak
    while (cvor.Previous != null)
        cvor = cvor.Previous;
    CvorDS temp = new CvorDS(a);
    if (n == 0) {
        temp.Previous = null;
        temp.Next = cvor;
        if (temp.Next != null)
        temp.Next.Previous = temp;
        return true;
    }
    int i = 0;
    while (i++ < n - 1) {
        // ako dodje do null pre n-te pozicije, staje jer ne moze da stavi na n-tu
        if (cvor.Next == null)
        return false;
        cvor = cvor.Next;
    }
    temp.Previous = cvor;
    temp.Next = cvor.Next;
    if (cvor.Next != null)
        cvor.Next.Previous = temp;
    cvor.Next = temp;
    return true;
}
```

Prebaci dati element DS liste na drugo/pretposlednje mesto u listi

Dat je pokazivac na neki cvor dvostruko spregnute liste koji je jedini ulazni podatak. Napisati funkciju koja ce ukazani element prebaciti na drugo/pretposlednje mesto u listi.

```
public static void moveToSecond(CvorDS cvor) {
    CvorDS iterator = cvor;
    while (iterator.Previous != null)
        iterator = iterator.Previous;
    // sada je iterator prvi element liste ako je prvi taj koji treba da se prebaci, samo zameni prvi i drugi
    if (iterator == cvor) {
        CvorDS prvi, drugi;
        prvi = iterator;
        drugi = iterator.Next;
        drugi.Previous = null;
        prvi.Next = drugi.Next;
        prvi.Previous = drugi;
        drugi.Next = prvi;
        if (prvi.Next != null)
            prvi.Next.Previous = prvi;
        return;
    }
    // izbacujemo cvor
    cvor.Previous.Next = cvor.Next;
    if (cvor.Next != null)
        cvor.Next.Previous = cvor.Previous;
    // ubacujemo cvor posle prvog
    cvor.Next = iterator.Next;
    cvor.Previous = iterator;
    if (iterator.Next != null)
```

```
        iterator.Next.Previous = cvor;
    iterator.Next = cvor;
}
```

Vrati razliku dva skupa preko DS liste

```
public boolean postoji (int podatak, CvorDS p) {
while (p != null) {
   if (p.podatak == podatak) return true;
   p= p.sledeci;
}
return false;
}

public CvorDS razlika2DS(CvorDS p1, CvorDS p2) {

CvorDS p3=null;
while(p1!=null) {
   if (!postoji(p1.podatak, p2) {
      p3= new CvorDS (p1.podatak, p3, null);
      if(p3.prethodni!=null) p3.prethodni.sledeci=p3;
   }
   p1= p1.sledeci;
}
return p3;
}
```

Vrati uniju dve DS liste

```
public boolean postoji (int podatak, CvorDS p) {
while (p != null) {
   if (p.podatak == podatak) return true;
   p= p.sledeci;
}
return false;
}

public CvorDS unija2DS (CvorDS p1, CvorDS p2) {
  CvorDS p3=null;
  while(p1 != null) {
     if (!postoji(p1.podatak, p2) {
        p3= new CvorDS (p1.podatak, p3, null);
        if(p3.prethodni!=null)
           p3.prethodni.sledeci=p3;
   }
   p1= p1.sledeci;
}
while (p2 != null) {
   p3= new CvorDS (p2.podatak, p3, null);
   if(p3.prethodni!=null)
      p3.prethodni.sledeci=p3;
   p2= p2.sledeci;
}
return p3;
}
```

Invertuj DS listu

```
public CvorDS invertujDS(CvorDS p1) {
    CvorDS p2=null;
    while(p1!=null){
        p2=new CvorDS (p1.podatak, null, p2);
        if (p2.sledeci != null)
            p2.sledeci.prethodni=p2;
        p1=p1.sledeci;
    }
    return p2;
}
```

Prebaci cvor DS liste na poslednje mesto. Nisu dati pokazivaci na poc i kraj liste.

```
public class CvorDS{
   CvorDS prethodni, sledeci;
   int broj;
}
public void prebaciNaKraj(CvorDS t) {
   CvorDS pom = t;
   while(pom.sledeci!=null) pom = pom.sledeci;//poslednji elem
   t.prethodni.sledeci = t.sledeci;
   t.sledeci.prethodni = t.prethodni;
   t.prethodni=pom;
   pom.sledeci=t;
   t.sledeci=null;
}
```

Prebaci poslednji cvor DS liste na prvo mesto. Dat je pok. na pocetak. Bez kreir. cvorova

```
publi class CvorDS{
CvorDS prethodni, sledeci;
int broj;

public void prebaciPoslNaPocetakDS(CvorDS prvi){
   CvorDS pom = prvi; //ovo nije novi cvor vec pokazivac na cvor
   while(pom.sledeci!=null) pom = pom.sledeci;
   pom.sledeci = prvi;
   prvi.prethodni = pom;
   pom.prethodni.sledeci = null;
   pom.prethodni = null;
   prvi = pom;
}
```

Prebaci cvor DS sa najvecom vred. na pocetak. Dat je pokazivac na kraj.

```
public void prebaciMax(CvorDS poslednji) {
        int maxpod = 0;
        CvorDS max = null;
        CvorDS pom = poslednji;
        while (pom!=null) {
          if(pom.pod>maxpod) {
                maxpod=pom.pod;
                max=pom;
          }
          pom = pom.prethodni;
        }
        max.prethodni.sledeci = max.sledeci;
        max.sledeci.prethodni = max.prethodni;
        max.sledeci=pom;//pom je sada prvi
        pom.prethodni=max;
        max.prethodni=null;
}
```

Dat je pokazivac na pocetak DS. Napisati metodu int **ZbirDuplih(CvorDSListe p1) koja ce izracunati zbir elemenata liste koji se javljaju u listi vise od jednom**. Na kraju operacije pocetna lista treba da ostane nepromenjena. (primer, ako lista ima elemente 2, 14, 5, 2, 5, 9 zbir je 2+5=7)

```
public int zbirDuplih (CvorDS p1){
   LinkedList<Integer> l = new LinkedList<Integer>(); //stack koji ce cuvati vrednosti brojeva ciji zbir trazimo
   int zbir=0;
   while(p1.sledeci!=null) {
      int elem=p1.pod;
      if(l.contains(elem) { //ako element postoji u steku idi dalje
         p1=p1.sledeci;
         continue;
      }
      CvorDS m = p1.sledeci;
      while(m!=null) { //proverava listu do kraja da vidi ima li element 'blizanca'
         if(m.pod== elem) { //ako naidje na isti element u DSListi ubacuje ga u l i izlazi iz petlje
            l.push(elem);
            break;
          }
         m=m. sledeci;
      }
      p1=p1. sledeci; //namesta p1 da pokazuje na sledeci element
  }
  while(!l.isEmpty) zbir+=l.pop();
  return zbir;
}
```

DS ciji cvorovi pokazuju na pocetak JS. Algoritam ubacivanja

Data je dvostruko spregnuta (DS) lista čiji su elementi čvorova pokazivači na početak jednostruko spregnute (JS) liste. Napisati klasu koja opisuje čvor ovakve DS liste, a zatim napisati algoritam za ubacivanje novog elementa u ovako definisanu strukturu, koji funkcioniše po sledećem principu: **kreće se od početka DS liste. Ako je element koji se ubacuje manji od prvog elementa JS liste trenutnog čvora DS liste, onda se taj element ubacuje na kraj te JS liste. U suprotnom, prelazi se na sledeći čvor DS liste i algoritam se ponavlja. Ako se stigne do kraja DS liste, onda se kreira novi čvor i u njegovu JS listu se ubacuje novi element.** Početna metoda prihvata pokazivač na početak DS liste i ceo broj koji se ubacuje.

```
public class CvorDS {
    CvorDS pret, sled;
    CvorJS prviJS;
}

public void ubaci(CvorDS prvi, int pod) {
    CvorDS pom = prvi;
    int broj = pod;
    bool ubacio = false;
    while(!ubacio && pom.sledeci!=null) {
        if(broj<pom.prviJS.pod) {
            dodajNaKrajJS(pom.PrviJS, broj);
            ubacio=true;
        } else {
            pom=pom.sled;
        }
    }
    if(ubacio=false) {
       dodajDS(pom, broj);
    }
}
public void dodajNaKrajJS(CvorJS cjs, int broj) {
    CvorJS pom = cjs;
    while (pom.sledeci!=null) pom = pom.sled;
    CvorJS novi = new CvorJS();
    novi.pod=broj;
    pom.sledeci=novi;
    novi.sledeci = null;
}
public void dodajDS(CvorDS poslednji, broj) {
    CvorDS novi = new CvorDS;
    poslednji.sledeci = novi;
    novi.sledeci = null;
    novi.prethodni = poslednji;

    CvorJS novijs= new CvorJS();
    novijs.pod = broj;
    novi.prviJS = novijs;
```

```
    novijs.sledeci = null;
}
```

Prikazi polovinu DS liste koja ima veci zbir. Dat je pokazivac na neki element liste.

Napisati **metodu koja prikazuje (stampa na ekranu) onu polovinu liste (levo od p ili desno od p) koja ima veci zbir**

```
private int zbirEllisteNapred(cvor p){
    int zbir=0;
    cvor pom=p;
    while(pom!=null){
        zbir+=pom.podatak;
        pom=pom.sledeci;
    }
    return zbir;
}

private int zbirElListeNazad(cvor p){
    int zbir=0;
    cvor pom=p;
    while(pom!=null){
        zbir+=pom.podatak;
        pom=pom.prethodni;
    }
    return zbir;
}

public void ispis(cvor p){
    cvor pom=p;
    while(pom!=null){
        System.out.println(pom.podatak);
        if(zbirEllisteNapred(p)>zbirElListeNazad(p))
            pom=pom.sledeci;
        else pom=pom.prethodni;
    }
}
```

Da li je string predstavljen DS listom palindrom. Dat je pokazivac na prvi.

```
public class CvorDS {
  char slovo;
  CvorDS prethodni;
  CvorDS sledeci;
}

public class DSLista {
CvorDS prvi;

public static boolean daLiJePalindromDS(CvorDS prvi) {
    CvorDS desno = prvi;
    CvorDS pom = prvi;
    while (pom.sledeci!=null) pom = pom.sledeci;
    CvorDS levo = pom; //poslednje slovo recenice, ako nema tacke na kraju
    while(levo != desno && levo.sledeci != desno) { //prvo je ako je neparan broj slova (bez razmaka), a
drugo za paran
      if(levo.slovo == ' ') //preskacu se razmaci, kao za npr 'rise to vote sir'
          levo = levo.prethodni;
      if(desno.slovo == ' ')
          desno = desno.sledeci;
      if(levo.slovo == desno.slovo) {
          levo = levo.prethodni;
          desno = desno.sledeci;
      } else
          return false;
    }
    return true;
  }
}
```

Vrati pokazivac na poslednji/pretposlednji/drugi element ciklicne DS liste

Dat je pokazivac na neki cvor DS **ciklicne** liste koja je sortirana u opadajucem redosledu. Napisati funkciju koja ce vratiti pokazivac na poslednji/pretposlednji/drugi element u listi.

```
public static CvorDS last(CvorDS cvor) {
    while (cvor.Next.Data < cvor.Data)
        cvor = cvor.Next;
    return cvor;
}
public static CvorDS pretposlednji(CvorDS cvor) {
    while (cvor.Next.Next.Data < cvor.Next.Data)
        cvor = cvor.Next;
    return cvor;
}
public static CvorDS second(CvorDS cvor) {
    while (cvor.Previous.Data > cvor.Data)
        cvor = cvor.Previous;
    return cvor.Next;
}
```

Vrati pokazivac na prvi/drugi element ciklicne DS liste

Dat je pokazivac na neki cvor DS **ciklicne** liste koja je sortirana u rastucem redosledu. Napisati funkciju koja ce vratiti pokazivac na prvi/drugi element u listi.

```
public static CvorDS first(CvorDS cvor) {
    while (cvor.Previous.Data < cvor.Data)
        cvor = cvor.Previous;
    return cvor;
}
```

STABLA

Dinamicka implementacija stabla.

```
public class Stablo implements binStablo {
private class Cvor {
    int info;
    Cvor levo;
    Cvor desno;
}
private Cvor koren;

private int visina (Cvor k) {
    if (k == null)
        return 0;
    else
        return  1+ Math.max(visina(k.levo), visina(k.desno));
 }

private void prefiks (Cvor k) {
    if (k == null)
        return;
    k.visit();
    prefiks (k.levo);
    prefiks (k.desno);
}

private void infiks (Cvor k) {
    if (k == null)
        return;
    infiks (k.levo);
    k.visit();
    infiks(k.desno);
}

private void postfiks (Cvor k) {
    if (k == null)
        return;
    postfiks (k.levo);
    posfiks(k.desno);
    k.visit();
}
```

Staticka implementacija stabla.

```
public class NizStablo implements binStablo {
private class Cvor {
  int info;
}
Cvor[] cvorovi;
int n;
}
public int visina() {
  return  floor (log(n)) + 1;  // n = 2^h-1
}
private void prefiks (int k) {
   if (k < 0 or k> n) return;
   cvorovi[k].visit();
   prefiks (k*2);  // prefiks levo
   prefiks (k*2+1); // prefiks desno
}
private void infiks (int k) {
   if (k < 0 or k> n) return;
   infiks (k*2);  // infiks levo
   cvorovi[k].visit();
   infiks (k*2+1); // infiks desno
}
private void postfiks (int k) {
   if (k < 0 or k> n) return;
   postfiks (k*2);  // postfiks levo
   postfiks (k*2+1); // postfiks desno
   cvorovi[k].visit();
}
```

Pokazivac na cvor binarnog stabla na najvecoj dubini

Dat je pokazivac na koren binarnog stabla ciji cvorovi sadrže cele brojeve. Napisati funkciju koja ce vratiti pokazivac na cvor list koji je na najvecoj dubini u stablu.

```
public static Cvor najdubljiCvor(Cvor koren) {
    if (koren == null)
        return null;
    /*ako je to list, znaci nasli smo ga*/
    if (koren.levo == null && koren.desno == null)
        return koren;
    /*ako nije to list, idi ka dubljem listu*/
    if (height(koren.levo) > height(koren.desno))
        return najdubljiCvor (koren.levo);
    else
        return najdubljiCvor (koren.desno);
}

private static int height(Cvor koren) {
```

```
    if (koren == null)
        return 0;
    return 1 + Math.max(height(koren.desno),
    height(koren.levo));
}
```

Pokazivac na cvor binarnog stabla na najmanjoj dubini

Dat je pokazivac na koren binarnog stabla ciji cvorovi sadrže cele brojeve. Napisati funkciju koja ce vratiti pokazivac na cvor list koji je na najmanjoj dubini u stablu.

```
public static Cvor najpliciCvor(Cvor koren) {
    if (koren == null)
        return null;
    /*ako je to list, znaci nasli smo ga*/
    if (koren.levo == null && koren.desno == null)
        return koren;
    /*ako nije to list, idi ka plicem listu*/
    if (firstLeafLevel(koren.levo) < firstLeafLevel(koren.desno))
        return najpliciCvor (koren.levo);
    else
        return najpliciCvor (koren.desno);
    }

    private static int firstLeafLevel(Cvor koren) {
        if (koren == null) /*na ovaj nacin ignorisu se cvorovi kojima je samo jedno dete null */
            return Integer.MAX_VALUE;
    /*ako je to list, nasli smo ga, vrati 0*/
    if (koren.levo == null && koren.desno == null)
        return 0;
    /*ako nije to list, trazi dalje*/
    return 1 + Math.min(firstLeafLevel(koren.desno), firstLeafLevel(koren.levo));
}
```

Pokazivac na roditelja datog cvora binarnog stabla

Napišite funkciju Cvor roditelj(cvor *k, cvor * p) koja prihvata pokazivac na koren binarnog stabla i pokazivac na neki cvor u stablu, a vraca pokazivac na roditelja cvora p (vraca NULL ako roditelj ne postoji). Pri tome cvor stabla ima samo pokazivace na svoju decu

```
public static Cvor roditelj(Cvor koren, Cvor p) {
    if (koren == null)
        return null;
    if (koren.levo != null) {
        if (areEqual(koren.levo, p))
            return koren;
        Cvor n = roditelj (koren.levo, p);
        if (n != null)
            return n;
    }
    if (koren.desno != null) {
```

```
        if (areEqual(koren.desno, p))
            return koren;
        Cvor n = roditelj(koren.desno, p);
        if (n != null)
            return n;
    }
    return null;
}
```

Pokazivac na roditelja datog cvora binarnog stabla – drugi nacin

Napišite funkciju Cvor roditelj(cvor *k, cvor * p) koja prihvata pokazivac na koren binarnog stabla i pokazivac na neki cvor u stablu, a vraca pokazivac na roditelja cvora p (vraca NULL ako roditelj ne postoji). Pri tome cvor stabla ima samo pokazivace na svoju decu

```
public static Cvor roditelj2(Cvor koren, Cvor p) {
    if (koren == null || koren == p)
        return null;
    if (koren.levo != null && koren.levo == p)
        return koren;
    if (koren.desno != null && koren.desno == p)
        return koren;
    Cvor parent = roditelj2 (koren.levo, p);
    if (parent == null)
        parent = roditelj2 (koren.desno, p);
    return parent;
}
```

Broj cvorova binarnog stabla koji su veci od svih svojih potomaka

Dat je pokazivac na koren binarnog stabla ciji cvorovi sadrže cele brojeve. Napišite funkciju koja ce vratiti broj cvorova koji su po sadržaju veci od sadržaja svih svojih potomoka.

```
public static int brojVecih(Cvor koren) {
    if (koren == null)
        return 0;
    if (koren.levo == null && koren.desno == null)
        return 0;// ignorisi listove
    int increment = 0;
    if (koren.data > max(koren.levo) && koren.data > max(koren.desno))
        increment = 1;
    return increment + brojVecih (koren.levo) + brojVecih (koren.desno);
}
private static int max(Cvor koren) {
    if (koren == null)
    return Integer.MIN_VALUE;
    return Math.max(koren.data, Math.max(max(koren.levo), max(koren.desno)));
}
```

Nivo datog cvora binarnog stabla

Napišite funkciju int nivo(cvor k, cvor p) koja prihvata pokazivac na koren binarnog stabla i pokazivac na neki cvor u stablu i vraca nivo na kome se pokazani cvor nalazi.

```
public static int nivo(Cvor koren, Cvor n, int level) {
    if (koren == null)
        return -1;
    if (koren == n)
        return level;
    int temp = nivo(koren.levo, n, level + 1);
    if (temp != -1)
        return temp;
    return nivo(koren.desno, n, level + 1);
}
```

Dublji cvor binarnog stabla od datog cvora

Napišite funkciju cvor nivo(cvor k, cvor p, cvor q) koja prihvata pokazivaè k na koren i pokazivace p i q na na neke cvorove u binarnom stablu i vraca pokazivac na onaj cvor koji se nalazi dublje u stablu (tj. ciji je nivo na kome se nalazi veci).

```
public static Cvor dubljiCvor(Cvor koren, Cvor a, Cvor b) {
    int nivoA = nivo(koren, a, 0);
    int nivoB = nivo (koren, b, 0);
    if (nivoA > nivoB)
        return a;
    else
        return b;
}
```

Cvor binarnog stabla kod koga je najmanji proizvod sadrzaja cvorova iz njegovog desnog podstabla

Dat je pokazivac na koren binarnog stabla ciji cvorovi sadrze cele brojeve. Napisati f-ju koja ce vratiti pokazivac na cvor u stablu kod koga je najmanji (najveci) proizvod sadrzaja cvorova iz njegovog desnog podstabla.

```
private static Cvor min;
private static int minValue;
public static Cvor minRightProduct(Cvor koren) {
    min = null;
    minValue = Integer.MAX_VALUE;
    findMinRightProduct(koren);
    return min;
}

private static void findMinRightProduct(Cvor koren) {
    if (koren == null)
        return;
    if (koren.desno == null) {
        findMinRightProduct(koren.levo);
    return;
    }
```

```
    int temp;
    if ((temp = product(koren.desno)) < minValue) {
        min = koren;
        minValue = temp;
    }
    findMinRightProduct(koren.desno);
    findMinRightProduct(koren.levo);
}
private static int product(Cvor koren) {
    if (koren == null)
        return 1;
    return koren.data * product(koren.desno) * product(koren.levo);
}
```

Provera da li su dva binarna stabla identična

```
public static boolean identicnaStabla(Cvor k1, Cvor k2) {
    if (k1 == null && k2 == null)
       return true;
    if (k1 != null && k2 != null) {
       if (k1.podatak == k2.podatak)
          return identicnaStabla(k1.levo, k2.levo) && identicnaStabla(k1.desno, k2.desno);
    }
    return false;
}
```

Provera da li su dva binarna stabla slicna kao u ogledalu

```
  public static boolean ogledalo(Cvor k1, Cvor k2){
    if (k1 == null && k2 == null)
        return true;
    if(k1!= null && k2!= null)
       return ogledalo(k1.levo, k2.desno) && ogledalo(k1.desno, k2.levo);
    return false;
}
```

Broj cvorova koji su veci od svih svojih potomaka

Dat je pokazivač na koren binarnog stabla čiji čvorovi sadrže cele brojeve. Napišite funkciju koja će vratiti broj čvorova koji su po sadržaju veći od sadržaja svih svojih potomoka

```
public static int potomci(Cvor t) {
  int p;
  if (t==null)
    return 0;
  if ((t.data>t.levo.data) && (t.data>t.desno.data))
    p=1;
  else p = 0;
  return p + potomci(t.levo) + potomci (t.desno);
}
```

Minimalna vrednost u cvoru stabla

```
public static int min(Cvor k) {
    if (k == null)
        return Integer.MAX_VALUE;
    return Math.min(k.pod, Math.min(min(k.levi), min(k.desni)));
}
```

Prebroj cvorove stabla

```
public static int Prebroj(Cvor c){
    if (c == null)
       return 0;
    return 1 + Prebroj (c.levo) + Prebroj (c.desno);
}
```

Zbir cvorova

```
public static int Zbir(Cvor c) {
  if(c == null)
     return 0;
  return c.pod + Zbir(c.levo) + Zbir(c.desno);
}
```

Broj cvorova sa sadrzajem vecim od zadatog broja

```
public static int izbrojVece(Cvor c, int v){
    if (c == null)
      return 0;
    if (c.podatak > v)
      return 1 + izbrojVece (c.levo, v) + izbrojVece (c.desno, v);
    else
      return izbrojVece (c.levo, v) + izbrojVece (c.desno, v);
}
```

Najmanji broj u binarnom stablu

```
public static int minimum(Cvor t){
    if (t == null)
      return Integer.MAX_VALUE;
    return Math.min(t.podatak, Math.min(minimum(t.levo), minimum(t.desno)));
}
```

Cvor u binarnom stablu koji sadrzi dati broj

```
public static Cvor nadji(Cvor t, int v){
    if (t == null)
      return null;
    if (t.podatak == v)
      return t;
    if (nadji(t.levo, v) != null) return nadji(t.levo, v);
    if (nadji(t.desno, v) != null) return nadji(t.desno, v);
}
```

Roditelj datog cvora

```
public static Cvor VratiRoditelja(Cvor koren, Cvor cvor)
{
   if (koren==null)||(koren==cvor) return null;
   if (koren.levi==cvor||koren.desni==cvor)
      return koren;
   if (VratiRoditelja(koren.levi, cvor)!=null) return VratiRoditelja(koren.levi, cvor);
   if (VratiRoditelja(koren.desni, cvor)!=null) return VratiRoditelja(koren.desni, cvor);
   return null;
}
```

Provera da li je cvor po sadrzaju veci od svojih potomaka

```
public static boolean veciOdPotomaka(Cvor t){
    if (t == null)
      return true;
    if (t.levo == null && t.desno == null)
      return true;
    if ((t.levo != null && t.podatak < t.levo.podatak) || (t.desno != null && t.podatak < t.desno.podatak))
      return false;
    return veciOdPotomaka(t.levo) && veciOdPotomaka(t.desno);
}
```

Broj cvorova u stablu koji zadovoljavaju neki uslov

```
public static int countIf(Cvor t){
    if (t == null)
        return 0;
    if (t.pod >5) //uslov
        return 1 + countIf(t.levo) + countIf(t.desno);
    return countIf(t.levo) + countIf(t.desno);
}
```

Broj cvorova koji su veci od potomaka

```
public static int countIf(Cvor t){
    if (t == null)
        return 0;
    if (VeciOdPotomaka(t)) //uslov
        return 1 + countIf(t.levo) + countIf(t.desno);
    return countIf(t.levo) + countIf(t.desno);
}
```

```
public static boolean veciOdPotomaka(Cvor t){
    if (t == null)
      return true;
    if (t.levo == null && t.desno == null)
      return true;
    if ((t.levo != null && t.podatak < t.levo.podatak) || (t.desno != null && t.podatak < t.desno.podatak))
      return false;
    return veciOdPotomaka(t.levo) && veciOdPotomaka(t.desno);
}
```

Vrati cvor kome je proizvod potomaka iz desnog podstabla najveci

```
public static Cvor najveciProizvod(Cvor t){
    if (t == null)
      return null;
    return veci(t, veci(najveciProizvod(t.levo), najveciProizvod(t.desno)));
}
public static Cvor veci(Cvor a, Cvor b){
    if (proizvodDesnogPodstabla(a) > proizvodDesnogPodstabla(b))
      return a;
    return b;
}
public static int proizvodDesnogPodstabla(Cvor t){
    if (t == null || t.desno == null)
      return 0;
    return proizvodClanova(t.desno);
}
public static int proizvodClanova(Cvor t){
    if (t == null)
      return 1;
    return t.podatak * proizvodClanova(t.levo) * proizvodClanova(t.desno);
}
```

Prosek elemenata desnog podstabla datog cvora

```
public static double Prosek(Cvor t) {
  if(t == null && t.desno ==null)
     return 0;
  int zbir = Zbir(t.desno);
  int brElemenata = Prebroj(t.desno);
  double prosek = zbir/brElemenata;
  return prosek;
}
public static int Zbir(Cvor c) {
  if(c == null)
     return 0;
  return c.pod + Zbir(c.levo) + Zbir(c.desno);
}
```

```
public static int Prebroj(Cvor c){
    if (c == null)
      return 0;
    return 1 + Prebroj (c.levo) + Prebroj (c.desno);
}
```

Prebrojava cvorove vece od 5

```
public int Count(Cvor c){
  if (c != null) {
    if (c > 5)
       return 1 + Count(c.levo) + Count(c.desno);
    else
       return Count(c.levo) + Count(c.desno);
  }
  return 0;
}
```

Prebrojava cvorove ciji je prosek veci od datog cvora

```
public int Count(Cvor t)
{
  if(t == null)
     return 0;
  if (Prosek(t) > t.Data)
     return 1 + Count(t.desno) + Count(t.levo);
  else
     return Count(t.desno) + Count(t.levo);
}
```

Uredjuje binarno stablo da cvor bude manji od njegove dece

Dat je pokazivac na koren binarnog stabla celih brojeva. Napisite algoritam koji ce dato stablo urediti tako da za svaki cvor vazi da je element u njemu manji od elemenata njegove dece (misli se samo na njegovo levo I desno dete).

```
public void Uredi(Cvor c)
{
   if(c==null)
      break;

   if(c.Data>c.levo.Data)
   {
      int pom=c.Data;
      c.Data=c.levo.Data;
      c.levo.Data=pom;
   }
   if(c.Data>c.desno.Data)
   {
      int pom=c.Data;
      c.Data=c.desno.Data;
      c.desno.Data=pom;
   }
   Uredi(c.levo);
```

```
    Uredi(c.desno);
}
```

Stampa cvorova stabla u opad. redosledu

```
public void ispisi(Cvor tek){
        if(tek==null) return;
        ispisi(tek.desno);
        System.out.println(tek.podatak);
        ispisi(tek.levo);
}
```

Stampa infix prolazak kroz stablo

```
public static void stampajInfix(Cvor k){
    if (k == null)
      return;
    stampajInfix(k.levo);
    System.out.println(k.podatak);
    stampajInfix(k.desno);
}
```

Stampa prefiks prolazak kroz stablo

```
public static void stampajStablo(Cvor k){
    if (k == null)
      return;
    System.out.println(k.podatak);
    stampajStablo(k.levo);
    stampajStablo(k.desno);
}
```

Stampa cvorove binarnog stabla na putu od korena do zadatog cvora

Ova metoda stampa sve cvorove na putu od korena BINARNOG STABLA do zadatog cvora. Dato stablo nije ni BST, ni AVL. Za to sluzi Stack u kome se cvorovi cuvaju, pa se na kraju stampaju.

```
public static void stampaOdKorenaDoCvora (Cvor koren, Cvor cvor) {
    stack = new Stack<Cvor>();
    boolean found = stampaOdKorenaDoCvora2(koren, cvor);
    if (!found) {
        System.out.println("Nije nadjen");
        return;
    }
    System.out.println(stack);
}
private static java.util.Stack<Cvor> stack;
private static boolean stampaOdKorenaDoCvora2 (Cvor koren, Cvor cvor) {
// ako im je bitan redosled stavi sve u Stack pa onda odstapmaj
    if (koren == null)
        return false;
    // stavi tekuci cvor na stek
    stack.add(koren);
    if (koren == cvor) {
        return true;
    }
    if (stampaOdKorenaDoCvora2 (koren.levo, cvor)) {
        return true;
    } else if (stampaOdKorenaDoCvora2 (koren.desno, cvor)) {
        return true;
    }
    stack.remove(koren);
    return false;
}
```

Stampa cvorove binarnog stabla od korena do cvora koji ima najmanju vrednost u stablu

Napisati proceduru koja štampa sadržaj svih cvorova binarnog stabla na putanji od korena do cvora koji ima najmanju vrednost u stablu.

```
public static void printToMin(Cvor koren) {
    if (koren == null)
        return;
    System.out.println(koren.data);
    if (min(koren) == koren.data)
        return;
    if (min(koren.levo) < min(koren.desno))
        printToMin(koren.levo);
    else
        printToMin(koren.desno);
}

private static int min(Cvor koren) {
```

```
    if (koren == null)
        return Integer.MAX_VALUE;
    return Math.min(koren.data, Math.min(min(koren.levo), min(koren.desno)));
}
```

Stampa cvorove binarnog stabla od korena do cvora koji ima najvecu vrednost u stablu

Napisati proceduru koja štampa sadržaj svih cvorova binarnog stabla na putanji od korena do cvora koji ima najvecu vrednost u stablu.

```
public static void printToMax(Cvor koren) {
    if (koren == null)
        return;
    System.out.println(koren.data);
    if (max(koren) == koren.data)
        return;
    if (max(koren.levo) > max(koren.desno))
        printToMax(koren.levo);
    else
        printToMax(koren.desno);
}
```

Stampa cvorove binarnog stabla na putu od zadatog cvora do korena

Ova metoda stampa sve cvorove na putu od zadatog cvora do korena BINARNOG STABLA.

```
public static boolean stampaOdCvoraDoKorena(Cvor koren, Cvor cvor) {
    if (koren == null)
        return false;
    if (koren.equals(cvor)) {
        System.out.println("-> " + koren.data);
        return true;
    }
    if (stampaOdCvoraDoKorena (koren.levo, cvor)) {
        System.out.println(koren.data);
        return true;
    }
    if (stampaOdCvoraDoKorena (koren.desno, cvor)) {
        System.out.println(koren.data);
        return true;
    }
    return false;
}
```

Stampa cvorove binarnog stabla na putanji od korena do tekuceg cvora

Dat je pokazivac na koren binarnog stabla ciji cvorovi sadrze cele brojeve i drugi pok na neki cvor u stablu. Napisati f-ju koja ce otstampati sve cvorove koji su na putanji od korena do datog cvora, ukljucujucu i ta dva cvora.

```
public static void StampajDoCvora(Cvor koren, Cvor cvor)
{
   Cvor tekuci=cvor;
   Cvor pom;
   if(VratiRoditelja(koren, tekuci)!=null) {
      pom = tekuci;
      StampajDoCvora(koren,VratiRoditelja(koren, tekuci);
      System.out.println(pom);
   }
   else return;
}

public static Cvor VratiRoditelja(Cvor koren, Cvor cvor)
{
   if (koren==null)||(koren==cvor) return null;
   if (koren.levi==cvor||koren.desni==cvor)
      return koren;
   if (VratiRoditelja(koren.levi, cvor)!=null) return VratiRoditelja(koren.levi, cvor);
   if (VratiRoditelja(koren.desni, cvor)!=null) return VratiRoditelja(koren.desni, cvor);
   return null;
}
```

Vraca najvecu vrednost cvora binarnog stabla

```
public static int najveci(Cvor c)
  {
    if(c==null)
        return Integer.MIN_VALUE;
    else
    {
        int levo = Math.max(c.pod, najveci(c.levo));
        int desno = Math.max(c.pod, najveci(c.desno));
        return (Math.max(levo, desno));
    }
}
```

Pretrazivanje BST – iterativno

```
public Cvor pretrazBST (int v, Cvor k) {
    while(k != null) {
        if (k.info > v)
            k = k.levo;
        else if (k.info < v)
            k = k.desno;
        else if (k.info == v)
            return k;
    }
    return null;
}
```

Pretrazivanje BST – rekurzivno

```
public Cvor pretraziBSTR (int v, Cvor k) {
    if (k == null)
        return null;
    if (k.info > v)
        return pretraziBSTR(v, k.levo)
    if (k.info < v)
        return pretraziBSTR (v, k.desno)
    else
        return k;
}
```

Provera da li je stablo AVL

```
public static boolean isAVL(Cvor c){
  return isBST(c) && isBalanced(c);
}
private static boolean isBST(Cvor c){
  if (c==null) return true;
  if ((c.levo.pod < c.pod)&&(c.desno.pod > c.pod))
    return isBST(c.levo) && isBST(c.desno);
  return false;
}
private static boolean isBalanced(Cvor c){
  if (c==null) return true;
  if (Math.abs((visina(c.levo)-visina(c.desno))<=1)
    return ((isBalanced(c.levo) && isBalanced(c.desno));
  return false;
}
private int visina (Cvor c) {
    if (c == null)
        return 0;
    else
        return  1+ Math.max(visina(c.levo), visina(c.desno));
 }
```

Broj cvorova koji ispunjavanju uslove za AVL stablo

Dat je pokazivac na koren binarnog stabla. Napišite funkciju koja ce vratiti broj cvorova koji ispunjavaju uslove za AVL stablo.

```
public static int howManyAVL(Cvor koren) {
    if (koren == null)
        return 0;
    int balanceFactor = height(koren.levo) - height(koren.desno);
    if (isBSTCvor(koren) && balanceFactor >= -1 && balanceFactor <= 1)
        return 1 + howManyAVL(koren.levo) + howManyAVL(koren.desno);
    else
        return howManyAVL (koren.levo) + howManyAVL(koren.desno);
}
private static int height(Cvor koren) {
    if (koren == null)
        return 0;
    return 1 + Math.max(height(koren.levo), height(koren.desno));
}
private static boolean isBSTCvor(Cvor koren) {
    boolean isBST = false;
    if (koren.levo != null) {
            if (koren.desno != null) {// oba !=null
                if (koren.data > koren.levo.data && koren.data < koren.desno.data)
                    isBST = true;
                } else {// left!=nill right=null
                    if (koren.data > koren.levo.data) isBST = true;
                }
            } else {// left=null
                if (koren.desno != null && koren.data < koren.desno.data)
                    isBST = true;
    else {// oba su null
        isBST = true;
    }
}
return isBST;
}
```

Broj cvorova koji ne ispunjavanju uslove za AVL stablo

Dat je pokazivac na koren binarnog stabla. Napišite funkciju koja ce vratiti broj cvorova koji ne ispunjavaju uslove za AVL stablo.

```
public static int howManyNonAVL(Cvor koren) {
    if (koren == null)
        return 0;
    int balanceFactor = height(koren.levo) - height(koren.desno);
    if (!(isBSTCvor(koren) && balanceFactor >= -1 && balanceFactor <= 1))
        return 1 + howManyNonAVL(koren.levo) + howManyNonAVL(koren.desno);
    else
        return howManyNonAVL(koren.levo) + howManyNonAVL(koren.desno);
```

```
}
```

Zbir putanje u AVL stablu

```
public static int zbirPutanje(Cvor a, Cvor b) {
    int zbir = 0;
    while (a.podatak != b.podatak) {
      zbir += a.podatak;
      if (a.podatak > b.podatak)
        a = a.levo;
      else if (a.podatak < b.podatak)
        a = a.desno;
    }
    zbir += b.podatak;
    return zbir;
}
```

Umetanje cvora u BST stablo - iterativno

```
public void insertBSTI(Cvor c) {
    Cvor tmp = this.koren;
    Cvor roditelj = null;
    while(tmp!=null) {
        roditelj=tmp;
        if(tmp.pod == c.pod) System.out.println("Cvor postoji");
        else if(c.pod < tmp.pod) tmp = tmp.levo;
        else tmp = tmp.desno;
    }
    if(roditelj.pod > c.pod) roditelj.levo = c;
    else roditelj.desno = c;
}
```

Umetanje cvora u BST stablo - rekurzivno

```
public void insertBSTR(Cvor c, Cvor koren) {
    if(koren==null) {koren = c; return;}
    if(c.pod == koren.pod) {
        System.out.println("Cvor postoji");
        return;
    }
    if(c.pod < koren.pod) insertBSTR(c, koren.levo);
    else insertBSTR(c, koren.desno);
}
```

AVL stablo – rotacija desno

```
public void rotirajDesno(Cvor x) {
    Cvor y = x.levo;
    Cvor tmp = y.desno;
    y.desno = x;
    x.levo = tmp;
```

```
}
```

AVL stablo – rotacija levo

```
public void rotirajLevo(Cvor x) {
    Cvor y = x.desno;
    Cvor tmp = y.levo;
    y.levo = x;
    x.desno = tmp;
}
```

Ubacuje element u BST i javlja da li je AVL

```
public Cvor insertInAVL(Cvor t, int data) {
        if (t == null)
           t = new Cvor(data);
        else {
           if (data < t.data) t.levo = insertInAVL(t.levo, data);
           if (data > t.data) t.desno = insertInAVL(t.desno, data);
        }

        if ((visina(t.levo) - visina(t.desno) == -2)) {  // da li je debalans na levoj strani
           if (t.desno.desno == null) {  //da li su potrebne dve rotacije
              t.desno = RotateRight(t.desno);
              return t = RotateLeft(t);
           } //zarotiraj prvo u desno "t.desno" pa onda u levo zarotiraj t
           else
              return t = RotateLeft(t); //zarotiraj "t" u levo
        }

        if ((visina(t.levo) - visina(t.desno) == 2)) {  // da li je debalans na desnoj strani
           if (t.levo.levo == null) {  //da li su potrebne dve rotacije
              t.levo = RotateLeft(t.levo);
              return t = RotateRight(t);
           } //zarotiraj prvo u levo "t.levo" pa onda u desno zarotiraj "t"
           else {
              return t = RotateRight(t);
           }  //zarotiraj "t" u desno
      }
      return t;
}
private Cvor RotateLeft(Cvor t) {
        Cvor noviKoren = t.desno;
        t.desno = noviKoren.levo;
        noviKoren.levo = t;
        return noviKoren;
}
```

```
private Cvor RotateRight(Cvor t) {
        Cvor noviKoren = t.levo;
        t.levo = noviKoren.desno;
        noviKoren.desno = t;
        return noviKoren;
}
```

Stampa sadrzaj svih cvorova AVL stabla u rastucem redosledu

Napisati funkciju koja prihvata pokazivac na koren AVL stabla i stampa sadrzaj svih cvorova stabla u rastucem redosledu.

```
public static void infix(Cvor koren) {
    // važi za svako BST, ne samo za AVL
    if (koren == null)
        return;
    infix (koren.levo);
    System.out.println(koren.data);
    infix (koren.desno);
}
```

Stampa sadrzaj svih cvorova AVL stabla u opadajucem redosledu

Napisati funkciju koja prihvata pokazivac na koren AVL stabla i stampa sadrzaj svih cvorova stabla u opadajucem redosledu.

```
public static void postfix(Cvor koren) {
    // važi za svako BST
    if (koren == null)
        return;
    postfix (koren.desno);
    System.out.println(koren.data);
    postfix (koren.levo);
}
```

Stampa cvorove na putanji od korena do datog cvora AVL stabla

Dat je pokazivac na koren AVL stabla ciji cvorovi sadrže cele brojeve i drugi pokazivac na neki cvor u stablu. Napisati funkciju koja ce odštampati sve cvorove koji su na putanji od korena do datog cvora.

```
public static void printRootToCvor(Cvor koren, Cvor cvor) {
    if (koren == null) {
        System.out.println("Not found");
        return;
    }
    if (koren.data == cvor.data) {
        System.out.println("found: " + koren.data);
        return;
    } else
        System.out.println(koren.data);
    if (koren.data > cvor.data)
        printRootToCvor(koren.levo, cvor);
    else
```

```
        printRootToCvor(koren.desno, cvor);
}
```

Stampa putanju u AVL stablu od cvora do cvora - iterativno

```
public static void stampajPutanju(Cvor a, Cvor b){
    if (a == null || b == null)
      return;
    while (a.podatak != b.podatak){
      System.out.println(a.podatak);
      if (a.podatak > b.podatak)
        a = a.levo;
      else
        a = a.desno;
    }
    System.out.println(b.podatak);
}
```

Stampa putanju u AVL stablu od cvora do cvora - rekurzivno

```
public static void stampajPutanjuR(Cvor a, Cvor b) {
    if (a==null || b==null)
      return;
    System.out.println(a.podatak);
    if (a.podatak > b.podatak)
      stampajPutanjuR(a.levo, b);
    if (a.podatak < b.podatak)
      stampajPutanjuR(a.desno, b);
}
```

Provera da li je stablo HEAP

```
public boolean IsHeap(Cvor k) {
   if (k==null) return true;
   if ((k.levo!=null)&&(k.levo.podatak>k.podatak)||!heap(k.levo))
      return false;
   if ((k.desno!=null)&&(k.desno.podatak>k.podatak)||!heap(k.desno))
      return false;
   return true;
}
```

HASH

Otvoreno adresiranje – hash tabela

```
public int HashSearch(HashTable[] Tablica, int kljuc) {
  for (int i=0; i<=Tablica.length; i++) {
      int j = hash(kljuc) + i;
      if (Tablica[j] == kljuc) {
          return j;
      }
      if (Tablica[j] == null) {
          break;
      }
  }

  return null;
}
```

Olancavanje

```
public int SearchInsertCh(HashTable[] Tablica, int key) {
    int i = hash(key)
    int j = 0;
    int free = 0;
    while (Tablica[i].key != key && Tablica[i].next != -1) {
        i = Tablica[i].next;
    }
    if(Tablica[i].key = key) return i;
    if(Tablica[i].key == null) {
        j = i;
    } else {
        while (Tablica[free].key != null) {
          free--;
        }
        j = free;
        Tablica[i].next = free;
   }
   Tablica[i].key = key;
   return 1;
}
```

Olancavanje – niz

Napisati funkciju koja implementira pretrazivanje niza celih brojeva koristeci transformaciju kljuca u adresu. Dimenzija niza je 102, a za funkciju transformacije koristiti moduo (ostatak celobrojnog deljenja). Problem kolizije se resava metodom olancavanja.

```
// h(k) = k mod niz.length, niz.length = 102, otv. adresiranje
public static int otvAdresiranje(int[] niz, int kljuc) {
  int j = kljuc%niz.length;
  if (niz[j] == kljuc)
      return niz[j];
  else {
      j ++;
      for (j; j<niz.length; j++) {
         if (niz[j] == kljuc) {
             return niz[j];
         } else
             return -1; //element nije nadjen
      }
 }
}
```