Zadaci

1. Koristeći primere priloženih klasa koje opisuju čvor binarnog stabla sa proizvoljnim podacima napraviti (ručno konstruisati) binarno stablo čiji podaci sadrže dva atributa: celobrojnu vrednost i njenu odgovarajuću vrednost u znakovnom obliku. Napraviti funkcije za dodavnje *levog* i *desnog* elementa proizvoljnom čvoru, kao i funkciju za ispis vrednosti proizvoljnog čvora.

2. Implementirati *InorderTreeWalk* funkciju. Pseudokod funkcije se nalazi na slici 1.

```
INORDER-TREE-WALK(x)
1  if x ≠ NIL
2      INORDER-TREE-WALK(x.left)
3      print x.key
4      INORDER-TREE-WALK(x.right)
```

Slika 1 – Pseudokod *InorderTreeWalk* funkcije

3. Implementirati *SearchTree* i *IterativeTreeSearch* funkcije. Pseudokodovi funkcija su priloženi na slici 2.

```
TREE-SEARCH(x, k)
1  if x == NIL or k == x.key
2      return x
3  if k < x.key
4      return TREE-SEARCH(x.left, k)
5  else return TREE-SEARCH(x.right, k)
```

```
ITERATIVE-TREE-SEARCH(x, k)
1  while x ≠ NIL and k ≠ x.key
2      if k < x.key
3          x = x.left
4      else x = x.right
5  return x
```

Slika 2 – Pseudokodovi *SearchTree* i *IterativeTreeSearch* funkcija

4. Implementirati *TreeMinimum*, *TreeMaximum* i *TreeSuccessor* funkcije. Pseudokodovi funkcija su priloženi na slici 3.

```
TREE-MINIMUM(x)
1  while x.left ≠ NIL
2      x = x.left
3  return x
```

```
TREE-MAXIMUM(x)
1  while x.right ≠ NIL
2      x = x.right
3  return x
```

```
TREE-SUCCESSOR(x)
1  if x.right ≠ NIL
2      return TREE-MINIMUM(x.right)
3  y = x.p
4  while y ≠ NIL and x == y.right
5      x = y
6      y = y.p
7  return y
```

Slika 3 – Pseudokodovi *TreeMinimum*, *TreeMaximum* i *TreeSuccessor* funkcija

5. Implementirati *TreeInsert* i *TreeDelete* funkciju. Pseudokodovi funkcija su priloženi na slici 4.

```
TREE-INSERT(T, z)
 1  y = NIL
 2  x = T.root
 3  while x ≠ NIL
 4        y = x
 5        if z.key < x.key
 6              x = x.left
 7        else x = x.right
 8  z.p = y
 9  if y == NIL
10        // tree T was empty
10        T.root = z
11  elseif z.key < y.key
12        y.left = z
13  else y.right = z
```

```
TREE-DELETE(T, z)
 1  if z.left == NIL
 2        TRANSPLANT(T, z, z.right)
 3  elseif z.right == NIL
 4        TRANSPLANT(T, z, z.left)
 5  else y = TREE-MINIMUM(z.right)
 6        if y.p ≠ z
 7              TRANSPLANT(T, y, y.right)
 8              y.right = z.right
 9              y.right.p = y
10        TRANSPLANT(T, z, y)
11        y.left = z.left
12        y.left.p = y
```

```
TRANSPLANT(T, u, v)
1  if u.p == NIL
2        T.root = v
3  elseif u == u.p.left
4        u.p.left = v
5  else u.p.right = v
6  if v ≠ NIL
7        v.p = u.p
```

Slika 4 – Pseudokodovi *TreeInsert, TreeDelete* i pomoćne *Transplant* funkcije

6. Napraviti proizvoljno dugačak niz slučajno generisanih celobrojnih vrednosti i iskoristiti ga kao ulaz za formiranje binarnog stabla. Izmeniti *InorderTreeWalk* funkciju da umesto ispisa elemente dodaje u listu. Slučajno generisani niz elemenata sortirati i proveriti rezultat. Za sortiranje se preporučuje upotreba funkcionalno proverenog algoritma.

Napomene:

- Proveriti funkcionalnost svih osnovnih funkcija.
- Za proveru osnovnih funkcija koristiti mali broj ulaznih podataka kako bi mogli ručno proveriti funkcionalnost i potvrditi ispravnost rada funkcije.