# Teorinė dalis (60%)

1. Duotas dvejetainis paieškos medis.

Iš eilės surašykite medžio elementų seką, kurią gautume eidami per medį tiesioginio apėjimo (angl. *predorder traversal*) būdu.

**Atsakymą rašykite čia:**

1. Duota elementų aibė yra realizuota dvejetainiu AVL paieškos medžiu, kuris atvaizduojamas tokiu būdu:

Kaip atrodys AVL paieškos medis pridėjus elementą 29? **Redaguokite pateiktą medį.**

1. Duota max heap struktūra:

Kaip atrodys ši struktūra, pašalinus iš jos maksimalų elementą? **Redaguokite pateiktą medį.**

1. Koks funkcijos E get(int k) asimptotinis laiko sudėtingumas?

private static class Node<E> {  
 private E element;   
 private Node<E> next;

Node(E data, Node<E> next) {  
 this.element = data;  
 this.next = next;  
 }  
public Node<E> findNode(int k) {  
 Node<E> e = this;  
 for (int i = 0; i < k; i++) {  
 e = e.next;  
 }  
 return e;  
 }  
}

private Node<E> first; // rodyklė į pirmą mazgą  
private Node<E> last; // rodyklė į paskutinį mazgą  
private Node<E> current; // rodyklė į einamąjį mazgą, naudojama getNext  
private int size; // sąrašo dydis, tuo pačiu elementų skaičius

public E get(int k) {  
 if (k < 0 || k >= size) {  
 return null;  
 }  
 current = first.findNode(k);  
 return current.element;  
}

**Atsakymą rašykite čia:**

1. Kokį rezultatą grąžins ši funkcija, jeigu parametras node yra vienkryptis sąrašas, susidedantis iš šių reikšmių: 3->6->1->7?

Node function(Node node)  
{  
 Node prev = null;  
 Node current = node;  
 Node next = null;  
   
 while (current != null) {  
 next = current.next;  
 current.next = prev;  
 prev = current;  
 current = next;  
 }  
   
 node = prev;  
 return node;  
}

**Atsakymą rašykite čia:**

1. Duota pradinė maišos lentelė, kurioje kolizijos sprendžiamos atskirų grandinėlių metodu. Naudodami dalybos maišos (angl. *division method of hashing*) funkciją (rakto reikšmės dalinamos iš lentelės talpos, kuri pradžioje yra lygi 8-iems), patalpinkite lentelėje raktus k = 22, 21, 53, 44, 18, 31, 2, 33. Lentelės parametrai: raktai – Integer tipo, apkrovimo faktorius – 0.75, permaišymo metu lentelė didinama 1.5 karto. Kaip atrodys tokia lentelė, įterpus visus aukščiau nurodytus raktus?

**Atsakymą rašykite čia:**

[0] ->...

[1] ->...

[2] ->...

[3] ->...

[4] ->...

[5] ->...

[6] ->...

[7] ->...

(Jeigu reikia, padidinkite masyvo talpą)

# Praktinė užduotis (40%)

Turime tokią steko duomenų struktūros realizacija.

import java.util.ArrayList;  
  
public class Stack<T>{  
 private ArrayList<T> stack;  
  
 public Stack(){  
 stack = new ArrayList<T>();  
 }  
  
 boolean isEmpty(){  
 return stack.size() == 0;  
 }  
  
 T max(){  
 throw new UnsupportedOperationException("Realizuokite operaciją, kuri grąžintų maksimalią steke esančią reikšmę. Operacijos asimptotinis sudėtingumas turi būti O(1).");  
 }  
  
 T peak(){

if(isEmpty()){  
 return null;  
 }

return stack.get(stack.size() - 1);  
 }  
  
 T pop(){

if(isEmpty()){  
 return null;  
 }

return stack.remove(stack.size() - 1);  
 }  
  
 void push(T item){  
 stack.add(item);  
 }  
  
 int size(){  
 return stack.size();  
 }  
  
 public static void main(String[] args) {  
 Stack<Integer> stack = new Stack<Integer>();  
  
 stack.push(0);  
 stack.push(10);  
 stack.push(3);  
 stack.push(-1);  
  
 System.*out*.println(stack.max());  
 }  
  
}

Reikia realizuoti metodą T max(), kuris grąžintų didžiausią steke esančią reikšmę. **Metodo asimptotinis sudėtingumas turėtų būti O(1).** Galima modifikuoti pateiktų steko operacijų kodus, tačiau negalima pakeisti jų asimptotinio laiko sudėtingumo bei veikimo logikos (modifikuotos operacijos rezultatas, turi sutapti su prieš modifikacija buvusios kodo versijos rezultatu).