JOBSHEET 14

Nama : Farid Aziz Wicaksono
Kelas : TI-1C
Absen : 14

A. Tugas

1. Tugas 1

Pada class BinarySearchTree, fungsi find() mengembalikan nilai bertipe boolean. Tambahkan fungsi baru yaitu findNode(), yang mengembalikan nilai bertipe Node.

```
1  package jobsheet14;
2
3  public class BinarySearchTree {
4
5      Node root;
6
7      public BinarySearchTree() {
8          root = null;
9      }
10
11     boolean isEmpty() {
12         return root == null;
13     }
14
15     void add(int data) {
16         if (isEmpty()) {
17             root = new Node(data);
18         } else {
19             Node current = root;
20             while (true) {
21                 if (data < current.data) {
22                     if (current.left != null) {
23                         current = current.left;
24                     } else {
25                         current.left = new Node(data);
26                         break;
27                     }
28                 } else if (data > current.data) {
29                     if (current.right != null) {
30                         current = current.right;
31                     } else {
32                         current.right = new Node(data);
33                         break;
34                     }
35                 } else {
36                     break;
37                 }
```

```
38          }
39      }
40
41      Node find(int data) {
42          Node temp;
43          Node current = root;
44          while (true) {
45              if (current.data == data) {
46                  temp = current;
47                  break;
48              } else if (data < current.data) {
49                  current = current.left;
50              } else {
51                  current = current.right;
52              }
53          }
54          return temp;
55      }
56
57      void traversepreorder(Node node) {
58          if (node != null) {
59              System.out.println(" " + node.data);
60              traversepreorder(node.left);
61              traversepreorder(node.right);
62          }
63      }
64
65      void traversepostorder(Node node) {
66          if (node != null) {
67              traversepostorder(node.left);
68              traversepostorder(node.right);
69              System.out.println(" " + node.data);
70          }
71      }
72
73      void traverseinorder(Node node) {
74          if (node != null) {
75              traverseinorder(node.left);
76              System.out.print(" " + node.data);
77              traverseinorder(node.right);
78          }
79      }
80
81      Node getsuccessor(Node del) {
            Node successor = del.right;
            Node successorParent = del;
```

```
82          while (successor.left != null) {
83              successorParent = successor;
84              successor = successor.left;
85          }
86          if (successor != del.right) {
87              successorParent.left = successor.right;
88              successor.right = del.right;
89          }
90          return successor;
91      }
92
93      public boolean delete(int id) {
94          Node parent = root;
95          Node current = root;
96          boolean isLeftChild = false;
97          while (current.data != id) {
98              parent = current;
99              if (current.data > id) {
100                 isLeftChild = true;
101                 current = current.left;
102             } else {
103                 isLeftChild = true;
104                 current = current.right;
105             }
106             if (current == null) {
107                 return false;
108             }
109         }
110         if (current.left == null && current.right == null) {
111             if (current == root) {
112                 root = null;
113             }
114             if (isLeftChild == true) {
115                 parent.left = null;
116             } else {
117                 parent.right = null;
118             }
119         } else if (current.right == null) {
120             if (current == root) {
121                 root = current.left;
122             } else if (isLeftChild) {
123                 parent.left = current.left;
124             } else {
125                 parent.right = current.left;
126             }
127         } else if (current.left == null) {
```

```
            if (current == root) {
                root = current.right;
            } else if (isLeftChild) {
                parent.left = current.right;
            } else {
                parent.right = current.right;
            }
        } else if (current.left != null && current.right != null) {
            Node successor = getsuccessor(current);
            if (current == root) {
                root = successor;
            } else if (isLeftChild) {
                parent.right = successor;
            }
            successor.left = current.left;
        }
        return true;
    }
}
```

2. Tugas 2

Pada class BinarySearchTree, buat fungsi getMax() untuk mencari node terbesar, dan getMin() untuk mencari node terkecil.

```
package jobsheet14;

public class BinarySearchTree {

    Node root;

    public BinarySearchTree() {
        root = null;
    }

    boolean isEmpty() {
        return root == null;
    }

    void add(int data) {
        if (isEmpty()) {
            root = new Node(data);
        } else {
            Node current = root;
            while (true) {
                if (data < current.data) {
                    if (current.left != null) {
                        current = current.left;
```

```
                } else {
                    current.left = new Node(data);
                    break;
                }
            } else if (data > current.data) {
                if (current.right != null) {
                    current = current.right;
                } else {
                    current.right = new Node(data);
                    break;
                }
            } else {
                break;
            }
        }
    }
}

Node find(int data) {
    Node temp;
    Node current = root;
    while (true) {
        if (current.data == data) {
            temp = current;
            break;
        } else if (data < current.data) {
            current = current.left;
        } else {
            current = current.right;
        }
    }
    return temp;
}

int Min() {
    Node current = root;
    while (true) {
        if (current.left != null) {
            current = current.left;
        } else {
            break;
        }
    }
    return current.data;
}

```

```
67      int Max() {
68          Node current = root;
69          while (true) {
70              if (current.right != null) {
71                  current = current.right;
72              } else {
73                  break;
74              }
75          }
76          return current.data;
77      }
78
79      void traversepreorder(Node node) {
80          if (node != null) {
81              System.out.println(" " + node.data);
82              traversepreorder(node.left);
83              traversepreorder(node.right);
84          }
85      }
86
87      void traversepostorder(Node node) {
88          if (node != null) {
89              traversepostorder(node.left);
90              traversepostorder(node.right);
91              System.out.println(" " + node.data);
92          }
93      }
94
95      void traverseinorder(Node node) {
96          if (node != null) {
97              traverseinorder(node.left);
98              System.out.print(" " + node.data);
99              traverseinorder(node.right);
100         }
101     }
102
103     Node getsuccessor(Node del) {
104         Node successor = del.right;
105         Node successorParent = del;
106         while (successor.left != null) {
107             successorParent = successor;
108             successor = successor.left;
109         }
110         if (successor != del.right) {
            successorParent.left = successor.right;
            successor.right = del.right;
```

```
111         }
112         return successor;
113     }
114 
115     public boolean delete(int id) {
116         Node parent = root;
117         Node current = root;
118         boolean isLeftChild = false;
119         while (current.data != id) {
120             parent = current;
121             if (current.data > id) {
122                 isLeftChild = true;
123                 current = current.left;
124             } else {
125                 isLeftChild = true;
126                 current = current.right;
127             }
128             if (current == null) {
129                 return false;
130             }
131         }
132         if (current.left == null && current.right == null) {
133             if (current == root) {
134                 root = null;
135             }
136             if (isLeftChild == true) {
137                 parent.left = null;
138             } else {
139                 parent.right = null;
140             }
141         } else if (current.right == null) {
142             if (current == root) {
143                 root = current.left;
144             } else if (isLeftChild) {
145                 parent.left = current.left;
146             } else {
147                 parent.right = current.left;
148             }
149         } else if (current.left == null) {
150             if (current == root) {
151                 root = current.right;
152             } else if (isLeftChild) {
153                 parent.left = current.right;
154             } else {
155                 parent.right = current.right;
156             }
```

```
157         } else if (current.left != null && current.right != null) {
158             Node successor = getsuccessor(current);
159             if (current == root) {
160                 root = successor;
161             } else if (isLeftChild) {
162                 parent.right = successor;
163             }
164             successor.left = current.left;
165         }
166         return true;
167     }
168 }
169
```

3. Tugas 3

   Modifikasi program diatas agar memiliki menu sebagai berikut:

   a. Insert

   b. Delete

   c. Find

   d. Display

   e. Keluar

```
1   package jobsheet14;
2   import java.util.Scanner;
3   public class BinarySearchTreeMain {
4       static void menu() {
5           System.out.println("=============");
6           System.out.println("= menu =");
7           System.out.println("1. Insert");
8           System.out.println("2. Delete");
9           System.out.println("3. Find");
10          System.out.println("4. Display");
11          System.out.println("5. keluar");
12          System.out.println("=============");
13      }
14
15      public static void main(String[] args) {
16          Scanner sc = new Scanner(System.in);
17          int pil = 0;
18          BinarySearchTree aa = new BinarySearchTree();
19          do {
20              menu();
21              System.out.print("Masukan Pilihan : ");
22              pil = sc.nextInt();
23              switch (pil) {
24                  case 1:
25                      System.out.println("Masukkan angka = ");
26                      int add = sc.nextInt();
```

```
                aa.add(add);
                break;
            case 2:
                System.out.println("Hapus = ");
                int id = sc.nextInt();
                aa.delete(id);
                break;
            case 3:
                System.out.print("Cari = ");
                int temu = sc.nextInt();
                Node cari = aa.find(temu);
                if (cari.data == temu) {
                    System.out.println("Ada");
                } else {
                    System.out.println("Tidak Ada");
                }
                break;
            case 4:
                aa.traverseinorder(aa.root);
                System.out.println("");
                aa.traversepreorder(aa.root);
                System.out.println("");
                aa.traversepostorder(aa.root);
                System.out.println("");
                break;
        }
    } while (pil != 5);
  }
}
```