

## Tugas 5

Nama : M irfan S

NRP 17111051

Source code

1.

```
1. public class BinaryTreeNode {
2.
3.     BinaryTreeNode parent;
4.     BinaryTreeNode left;
5.     BinaryTreeNode right;
6.     int data;
7.
8.     BinaryTreeNode(int new_data) {
9.         this.data = new_data;
10.        this.parent = null;
11.        this.left = null;
12.        this.right = null;
13.    }
14.
15.
16.    void set_parent(BinaryTreeNode other) {
17.        this.parent = other;
18.        if (other != null) {
19.            if (other.data > this.data) {
20.                other.left = this;
21.            } else {
22.                other.right = this;
23.            }
24.        }
25.    }
26.
27.    void set_left(BinaryTreeNode other) {
28.        this.left = other;
29.        if (other != null) {
30.            other.parent = this;
31.        }
32.    }
33.
34.    void set_right(BinaryTreeNode other) {
35.        this.right = other;
36.        if (other != null) {
37.            other.parent = this;
38.        }
39.    }
40.    boolean is_left() {
41.        return this.parent != null && parent.left == this;
42.    }
43.
44.    boolean is_right() {
45.        return this.parent != null && parent.right == this;
46.    }
47.
48.    boolean has_right_and_left() {
49.        return this.left != null && this.right != null;
```

```

50.     }
51.
52.     boolean only_has_left() {
53.         return this.left != null && this.right == null;
54.     }
55.
56.     boolean only_has_right() {
57.         if (this.right != null || this.left == null) {
58.
59.         }
60.         return this.right != null && this.left == null;
61.     }
62.
63.     boolean has_no_child() {
64.         return this.left == null && this.right == null;
65.     }
66.     void unset_parent() {
67.         if (this.is_left()) {
68.             parent.left = null;
69.             this.parent = null;
70.
71.         } else if (this.is_right()) {
72.             parent.right = null;
73.             this.parent = null;
74.
75.         }
76.     }
77.
78.
79.     BinaryTreeNode most_left_child() {
80.         BinaryTreeNode child = this.left;
81.         while (child.left != null) {
82.             child = child.left;
83.
84.         }
85.         return child;
86.     }
87.
88.     BinaryTreeNode most_right_child() {
89.         BinaryTreeNode child = this.right;
90.         while (child.right != null) {
91.             child = child.right;
92.         }
93.         return child;
94.     }
95.
96.     void print(String spaces, String label) {
97.         System.out.println(spaces + label + this.data);
98.         if (this.left != null) {
99.             this.left.print(spaces + " ", " LEFT ");
100.        }
101.        if (this.right != null) {
102.            this.right.print(spaces+ " ", " RIGHT ");
103.        }
104.    }
105.
106.    void print() {
107.        this.print(" ", "NODE ");
108.    }
109.
110.    void infix() {
111.        System.out.print("(" );
112.        if (this.left != null) {
113.            left.infix();
114.        } else {
115.            System.out.print("null");

```

```

116.         }
117.         System.out.print(" " + this.data + " ");
118.         if (this.right != null) {
119.             right.infix();
120.         } else {
121.             System.out.print("null");
122.         }
123.         System.out.print("\n");
124.     }
125.
126.     void prefix() {
127.         System.out.print(this.data + "(");
128.         if (this.left != null) {
129.             left.prefix();
130.         } else {
131.             System.out.print("null");
132.         }
133.         System.out.print(" ");
134.         if (this.right != null) {
135.             right.prefix();
136.         } else {
137.             System.out.print("null");
138.         }
139.         System.out.print(") ");
140.     }
141.     void postfix() {
142.         System.out.print("(" + " ");
143.         if (this.left != null) {
144.             left.postfix();
145.         } else {
146.             System.out.print("null");
147.         }
148.         System.out.print(" ");
149.         if (this.right != null) {
150.             right.postfix();
151.         } else {
152.             System.out.print("null");
153.         }
154.         System.out.print(")" + this.data);
155.     }
156. }

```

2.

```

1. import java.util.Scanner;
2.
3. public class search {
4.
5.     public static void main(String[] args) {
6.         BinaryTree bt = new BinaryTree();
7.         Scanner sc = new Scanner(System.in);
8.         int angka = 0, jumangka, cari;
9.         char ulang = 'y';
10.
11.         System.out.println("    ** Binary Search dalam Binary Tree **");
12.         System.out.println("-----");
13.         System.out.print("Masukan jumlah angka\t: ");
14.         jumangka = sc.nextInt();
15.         for (int i = 0; i < jumangka; i++) {
16.             System.out.print("Angka ke " + (i + 1) + "\t: ");
17.             angka = sc.nextInt();
18.             bt.push(new BinaryTreeNode(angka));
19.         }

```

```

20.     System.out.println("-----");
21.     bt.print();
22.     do {
23.         System.out.println("-----");
24.         System.out.print("Masukan angka yang anda cari : ");
25.         cari = sc.nextInt();
26.         bt.caricari(cari);
27.         do {
28.             System.out.print("Cari Angka lagi? (Y / T)\t");
29.             ulang = sc.next().charAt(0);
30.         } while (ulang != 't' && ulang != 'y');
31.     } while (ulang == 'y');
32. }
33.
34. }
35.
36. }

```

3.

```

1. public class BinaryTree {
2.
3.     BinaryTreeNode root;
4.
5.     public BinaryTree() {
6.         this.root = null;
7.     }
8.
9.     void print() {
10.        if (this.root != null) {
11.            this.root.print();
12.        }
13.    }
14.
15.    void prefix() {
16.        if (this.root != null) {
17.            this.root.prefix();
18.        }
19.        System.out.println("");
20.    }
21.
22.    void infix() {
23.        if (this.root != null) {
24.            this.root.infix();
25.        }
26.        System.out.println("");
27.    }
28.    void postfix() {
29.        if (this.root != null) {
30.            this.root.postfix();
31.        }
32.        System.out.println("");
33.    }
34.    void push(BinaryTreeNode new_node) {
35.        if (this.root == null) {
36.            this.root = new_node;
37.        } else {
38.            BinaryTreeNode current = this.root;
39.            while (current != null) {
40.                if (new_node.data > current.data) {
41.                    if (current.right == null) {
42.                        current.set_right(new_node);
43.                        break;

```

```

44.         } else {
45.             current = current.right;
46.         }
47.     } else {
48.         if (current.left == null) {
49.             current.set_left(new_node);
50.             break;
51.         } else {
52.             current = current.left;
53.         }
54.     }
55. }
56. }
57. }
58.
59. void delete(BinaryTreeNode deleted) {
60.     if (this.root != null) {
61.         if (deleted.has_no_child()) {
62.             if (deleted == this.root) {
63.                 this.root = null;
64.             } else {
65.                 deleted.unset_parent();
66.             }
67.         } else if (deleted.only_has_left() || deleted.only_has_right()) {
68.             BinaryTreeNode replacement = null;
69.             if (deleted.only_has_left()) {
70.                 replacement = deleted.left;
71.             } else {
72.                 replacement = deleted.right;
73.             }
74.             if (deleted == this.root) {
75.                 this.root = replacement;
76.                 this.root.unset_parent();
77.             }
78.             } else if (deleted.is_left()) {
79.                 deleted.parent.set_left(replacement);
80.                 deleted.unset_parent();
81.             }
82.             } else if (deleted.is_right()) {
83.                 deleted.parent.set_right(replacement);
84.                 deleted.unset_parent();
85.             }
86.         }
87.     } else {
88.         BinaryTreeNode replacement = deleted.left;
89.         if (replacement.right != null) {
90.             replacement = replacement.most_right_child();
91.         }
92.         BinaryTreeNode parent_of_replacement = replacement.parent;
93.         if (replacement.only_has_right()) {
94.             parent_of_replacement.set_left(replacement.right);
95.         }
96.         replacement.unset_parent();
97.         replacement.set_left(deleted.left);
98.         replacement.set_right(deleted.right);
99.         if (deleted == this.root) {
100.             this.root = replacement;
101.         } else if (deleted.is_left()) {
102.             deleted.parent.set_left(replacement);
103.         } else if (deleted.is_right()) {
104.             deleted.parent.set_right(replacement);
105.         }
106.     }
107. }
108. }
109.

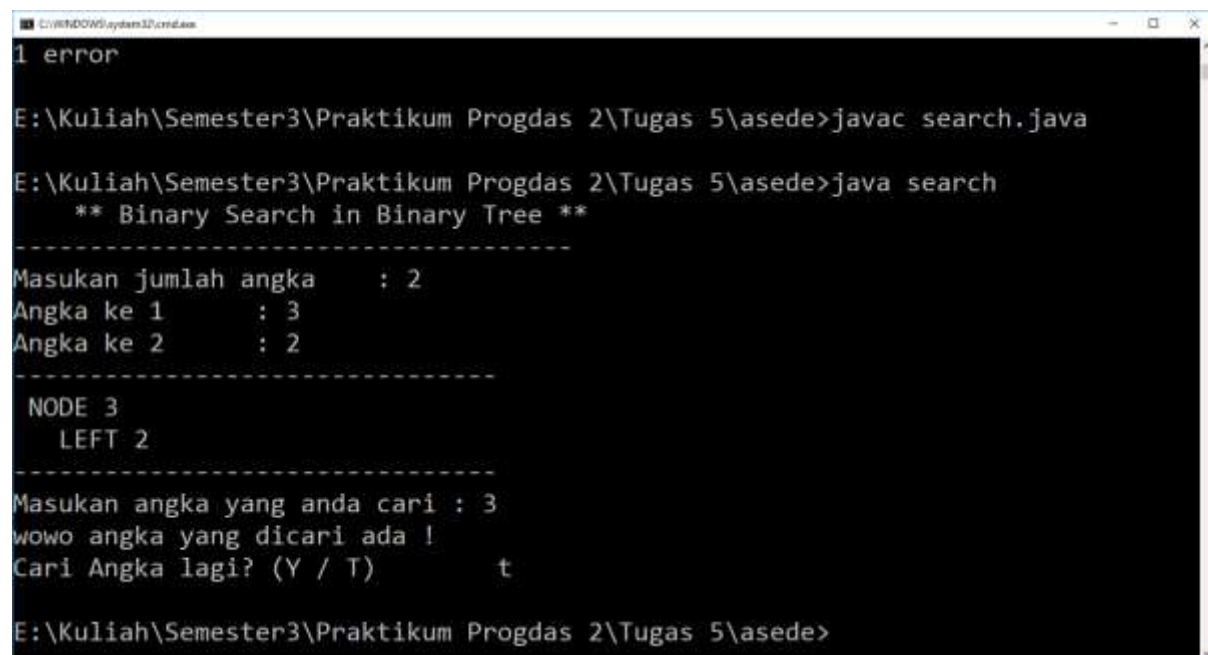
```

```

110.         void caricari(int key) {
111.             if (this.root == null) {
112.                 System.out.println("kosong ");
113.             } else {
114.                 BinaryTreeNode current = this.root;
115.                 while (current != null) {
116.                     if (key == current.data) {
117.                         System.out.println("Angka tersebut ada");
118.                         break;
119.                     }
120.                     if (key > current.data) {
121.                         current = current.right;
122.                     } else {
123.                         current = current.left;
124.                     }
125.                 }
126.             }
127.         }
128.     }
129.
130. }

```

Output:



```

C:\WINDOWS\system32\cmd.exe
1 error

E:\Kuliah\Semester3\Praktikum Progdas 2\Tugas 5\asede>javac search.java

E:\Kuliah\Semester3\Praktikum Progdas 2\Tugas 5\asede>java search
    ** Binary Search in Binary Tree **
-----
Masukan jumlah angka      : 2
Angka ke 1                : 3
Angka ke 2                : 2
-----
NODE 3
  LEFT 2
-----
Masukan angka yang anda cari : 3
wowo angka yang dicari ada !
Cari Angka lagi? (Y / T)      t

E:\Kuliah\Semester3\Praktikum Progdas 2\Tugas 5\asede>

```