**Tugas 5**

Nama : M irfan S

NRP 17111051

Source code

1.

1. **public** **class** BinaryTreeNode {
3. BinaryTreeNode parent;
4. BinaryTreeNode left;
5. BinaryTreeNode right;
6. **int** data;
8. BinaryTreeNode(**int** new\_data) {
9. **this**.data = new\_data;
10. **this**.parent = **null**;
11. **this**.left = **null**;
12. **this**.right = **null**;
13. }

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. }
27. **void** set\_left(BinaryTreeNode other) {
28. **this**.left = other;
29. **if** (other != **null**) {
30. other.parent = **this**;
31. }
32. }
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. }
44. **boolean** is\_right() {
45. **return** **this**.parent != **null** && parent.right == **this**;
46. }
48. **boolean** has\_right\_and\_left() {
49. **return** **this**.left != **null** && **this**.right != **null**;
50. }
52. **boolean** only\_has\_left() {
53. **return** **this**.left != **null** && **this**.right == **null**;
54. }
56. **boolean** only\_has\_right() {
57. **if** (**this**.right != **null** || **this**.left == **null**) {
59. }
60. **return** **this**.right != **null** && **this**.left == **null**;
61. }
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**;
71. } **else** **if** (**this**.is\_right()) {
72. parent.right = **null**;
73. **this**.parent = **null**;
75. }
76. }

79. BinaryTreeNode most\_left\_child() {
80. BinaryTreeNode child = **this**.left;
81. **while** (child.left != **null**) {
82. child = child.left;
84. }
85. **return** child;
86. }
88. BinaryTreeNode most\_right\_child() {
89. BinaryTreeNode child = **this**.right;
90. **while** (child.right != **null**) {
91. child = child.right;
92. }
93. **return** child;
94. }
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. }
106. **void** print() {
107. **this**.print(" ", "NODE ");
108. }
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(")");
124. }
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;
3. **public** **class** search {
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';
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');
32. } **while** (ulang == 'y');
34. }
36. }

3.

1. **public** **class** BinaryTree {
3. BinaryTreeNode root;
5. **public** BinaryTree() {
6. **this**.root = **null**;
7. }
9. **void** print() {
10. **if** (**this**.root != **null**) {
11. **this**.root.print();
12. }
13. }
15. **void** prefix() {
16. **if** (**this**.root != **null**) {
17. **this**.root.prefix();
18. }
19. System.out.println("");
20. }
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. }
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();
78. } **else** **if** (deleted.is\_left()) {
79. deleted.parent.set\_left(replacement);
80. deleted.unset\_parent();
82. } **else** **if** (deleted.is\_right()) {
83. deleted.parent.set\_right(replacement);
84. deleted.unset\_parent();
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. }
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. }
128. }
130. }

Output:

