Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

Kelas : KOMB1

Tugas 5 – Heap Sort

Source Code

```
1   public class Main {
2       public static void main(String[] args) {
3           int[] key = {78, 3, 9, 10, 23, 77, 34, 86, 90, 100, 20,
    66, 94, 63, 97};
4
5           Heap heap = new Heap(15);
6
7           for(int i = 0; i < key.length; i++) {
8               heap.insert(key[i]);
9           }
10
11          heap.heapSort();
12      }
13  }
14
15  class Node {
16      private int data;
17
18      public Node(int d) {
19          this.data = d;
20      }
21
22      public int getData() {
23          return this.data;
24      }
25
26      public void setData(int d) {
27          this.data = d;
28      }
29  }
30
31  class Heap {
32      private Node[] array;
33      private int maxSize, currentSize;
34
35      public Heap(int max) {
36          maxSize = max;
37          currentSize = 0;
```

```
38          array = new Node[maxSize + 1];
39      }
40
41      public boolean isFull() {
42          return currentSize == maxSize;
43      }
44
45      public boolean hasLeftChild(int i) {
46          return 2 * i <= currentSize;
47      }
48
49      public boolean hasRightChild(int i) {
50          return 2 * i + 1 <= currentSize;
51      }
52
53      public boolean insert(int data) {
54          if(isFull()) {
55              return false;
56          }
57
58          array[++currentSize] = new Node(data);
59          trickleUp(currentSize);
60
61          return true;
62      }
63
64      public void trickleUp(int i) {
65          int parent = i / 2;
66          Node bottom = array[i];
67
68          while(i > 1 && array[parent].getData() <
    bottom.getData()) {
69              array[i] = array[parent];
70              i = parent;
71              parent = i / 2;
72          }
73
74          array[i] = bottom;
75      }
76
77      public Node remove() {
78          Node root = array[1];
79          array[1] = array[currentSize--];
80          trickleDown(1);
81
82          return root;
83      }
84
```

```
85      public void trickleDown(int i) {
86          Node  top = array[i];
87          int largerChild;
88
89          while (hasLeftChild(i)) {
90              int leftChild = 2 * i;
91              int rightChild = leftChild + 1;
92
93              if (hasRightChild(i) && array[rightChild].getData() >
    array[leftChild].getData()) {
94                  largerChild = rightChild;
95              }
96              else {
97                  largerChild = leftChild;
98              }
99
100             if (top.getData() >= array[largerChild].getData()) {
101                 break;
102             }
103
104             array[i] = array[largerChild];
105             i = largerChild;
106         }
107
108         array[i] = top;
109     }
110
111     public void heapSort() {
112         for(int i = currentSize; i >= 1; i--) {
113             Node max = remove();
114             array[currentSize + 1] = max;
115         }
116         displayHeap();
117     }
118
119     public void displayHeap() {
120         System.out.println();
121
122         System.out.println("Sorted Heap Tree (Ascending): ");
123         for(int i = 1; i <= maxSize; i++) {
124             System.out.print(array[i].getData() + " ");
125         }
126
127         System.out.println("\n");
128
129         System.out.println("Sorted Heap Tree (Descending): ");
130         for(int i = maxSize; i >= 1; i--) {
131             System.out.print(array[i].getData() + " ");
```

```
132         }
133
134         System.out.println("\n");
135     }
136 }
```

Output Terminal

```
Sorted Heap Tree (Ascending):
3 9 10 20 23 34 63 66 77 78 86 90 94 97 100

Sorted Heap Tree (Descending):
100 97 94 90 86 78 77 66 63 34 23 20 10 9 3
```