Deccan Education Society's (DES)
Pune University, Pune
School of Engineering and Technology
Department of Computer Engineering and Technology
Program: B. Tech in Computer Science and Engineering

Academic Year: 2024-25	**Year:** Third Year	**Term:** II
PRN No.: 1012412079	**Name: Ratnajeet Patil**	
Subject: DS2		
Assignment No.: 2		
Date:		

Lab Assignment: 03

Title: Write a modular program to implement primitive operations on Min and Max Heap.

Code:

```java
import java.util.*;
// Write a modular program to implement primitive operations on Min/Max Heap.

import java.util.*;

class HeapStructure {
    int[] heap;
    int size;
    int maxSize;
    boolean isMinHeap;

    HeapStructure(int maxSize, boolean isMinHeap) {
        this.maxSize = maxSize;
        this.size = 0;
        this.isMinHeap = isMinHeap;
        heap = new int[this.maxSize + 1];
        heap[0] = isMinHeap ? Integer.MIN_VALUE : Integer.MAX_VALUE;
    }

    int parent(int pos) {
        return pos / 2;
    }

    int leftChild(int pos) {
        return (2 * pos);
    }

    int rightChild(int pos) {
        return (2 * pos) + 1;
```

```
30.     }
31.
32.     boolean isLeaf(int pos) {
33.         return pos >= (size / 2) && pos <= size;
34.     }
35.
36.     void swap(int fpos, int spos) {
37.         int tmp = heap[fpos];
38.         heap[fpos] = heap[spos];
39.         heap[spos] = tmp;
40.     }
41.
42.     void heapify(int pos) {
43.         if (!isLeaf(pos)) {
44.             if (isMinHeap) {
45.                 minHeapify(pos);
46.             } else {
47.                 maxHeapify(pos);
48.             }
49.         }
50.     }
51.
52.     void minHeapify(int pos) {
53.         if (!isLeaf(pos)) {
54.             int leftPos = leftChild(pos);
55.             int rightPos = rightChild(pos);
56.             int smallestPos = pos;
57.
58.             if (leftPos <= size && heap[leftPos] < heap[smallestPos]) {
59.                 smallestPos = leftPos;
60.             }
61.             if (rightPos <= size && heap[rightPos] < heap[smallestPos]) {
62.                 smallestPos = rightPos;
63.             }
64.
65.             if (smallestPos != pos) {
66.                 swap(pos, smallestPos);
67.                 minHeapify(smallestPos);
68.             }
69.         }
70.     }
71.
72.     void maxHeapify(int pos) {
73.         if (!isLeaf(pos)) {
74.             int leftPos = leftChild(pos);
75.             int rightPos = rightChild(pos);
76.             int largestPos = pos;
77.
78.             if (leftPos <= size && heap[leftPos] > heap[largestPos]) {
79.                 largestPos = leftPos;
80.             }
81.             if (rightPos <= size && heap[rightPos] > heap[largestPos]) {
82.                 largestPos = rightPos;
83.             }
84.
85.             if (largestPos != pos) {
86.                 swap(pos, largestPos);
87.                 maxHeapify(largestPos);
88.             }
89.         }
90.     }
```

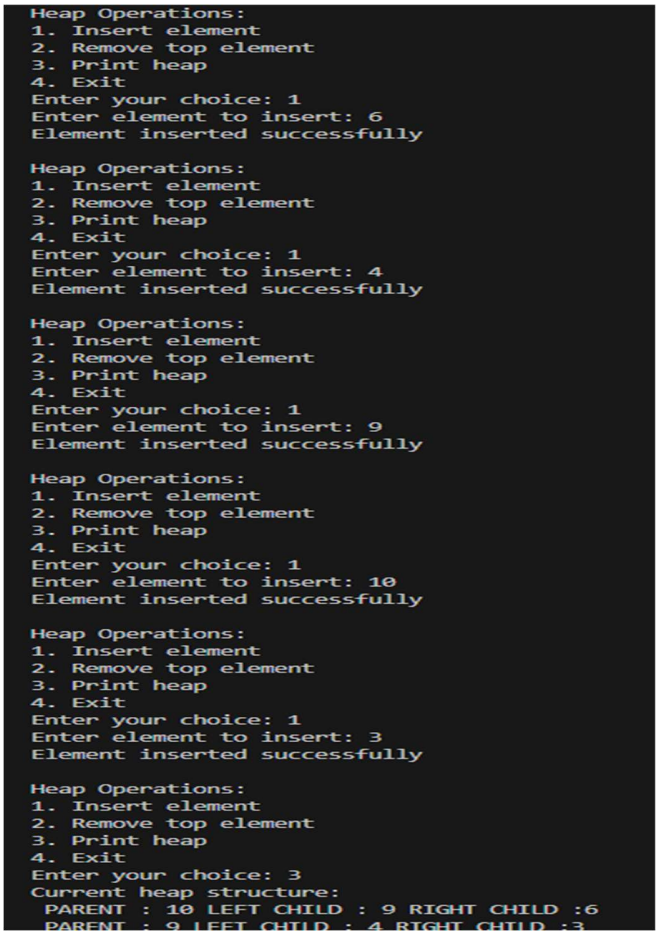
```
91.
92.     void insert(int element) {
93.         if (size >= maxSize) {
94.             System.out.println("Heap is full!");
95.             return;
96.         }
97.
98.         heap[++size] = element;
99.         int current = size;
100.
101.        if (isMinHeap) {
102.            while (heap[current] < heap[parent(current)]) {
103.                swap(current, parent(current));
104.                current = parent(current);
105.            }
106.        } else {
107.            while (heap[current] > heap[parent(current)]) {
108.                swap(current, parent(current));
109.                current = parent(current);
110.            }
111.        }
112.    }
113.
114.    void print() {
115.        if (size == 0) {
116.            System.out.println("Heap is empty!");
117.            return;
118.        }
119.
120.        for (int i = 1; i <= size / 2; i++) {
121.            System.out.print(" PARENT : " + heap[i] + " LEFT CHILD : " + heap[2 * i]);
122.            if (2 * i + 1 <= size) {
123.                System.out.print(" RIGHT CHILD :" + heap[2 * i + 1]);
124.            }
125.            System.out.println();
126.        }
127.    }
128.
129.    void buildHeap() {
130.        for (int pos = (size / 2); pos >= 1; pos--) {
131.            heapify(pos);
132.        }
133.    }
134.
135.    int remove() {
136.        if (size == 0) {
137.            System.out.println("Heap is empty!");
138.            return -1;
139.        }
140.
141.        int popped = heap[1];
142.        heap[1] = heap[size--];
143.        heapify(1);
144.        return popped;
145.    }
146. }
147.
148. public class HeapImplementation {
149.     public static void main(String[] arg) {
150.         Scanner scanner = new Scanner(System.in);
151.         System.out.println("Enter max size of heap: ");
```

```
        int maxSize = scanner.nextInt();

        System.out.println("Select type of heap:");
        System.out.println("1. Min Heap");
        System.out.println("2. Max Heap");
        int heapType = scanner.nextInt();

        boolean isMinHeap = heapType == 1;
        HeapStructure heapObj = new HeapStructure(maxSize, isMinHeap);

        int choice;
        do {
            System.out.println("\nHeap Operations:");
            System.out.println("1. Insert element");
            System.out.println("2. Remove top element");
            System.out.println("3. Print heap");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter element to insert: ");
                    int element = scanner.nextInt();
                    heapObj.insert(element);
                    System.out.println("Element inserted successfully");
                    break;

                case 2:
                    int removed = heapObj.remove();
                    if (removed != -1) {
                        System.out.println("Removed element: " + removed);
                    }
                    break;

                case 3:
                    System.out.println("Current heap structure:");
                    heapObj.print();
                    break;

                case 4:
                    System.out.println("Exiting program...");
                    break;

                default:
                    System.out.println("Invalid choice! Please try again.");
            }

        } while (choice != 4);

        scanner.close();
    }
}
```

Min Heap:

```
Enter max size of heap:
6
Select type of heap:
1. Min Heap
2. Max Heap
1

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 5
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 4
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 3
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 8
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 3
Current heap structure:
 PARENT : 3 LEFT CHILD : 5 RIGHT CHILD :4
 PARENT : 5 LEFT CHILD : 8
```

Max Heap:



```
Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 6
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 4
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 9
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 10
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 1
Enter element to insert: 3
Element inserted successfully

Heap Operations:
1. Insert element
2. Remove top element
3. Print heap
4. Exit
Enter your choice: 3
Current heap structure:
 PARENT : 10 LEFT CHILD : 9 RIGHT CHILD :6
 PARENT : 9 LEFT CHILD : 4 RIGHT CHILD :3
```