



Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of AI-ML

Experiment	10
Aim	Implement the given problem statement
Objective	Heap operations - Implement operations of heap structures
Name	Balla Mahadev Shrikrishna
UCID	2023300010
Class	A
Batch	A
Date of Submission	30-10-24

Explanation of the technique used

1 -

Q. Construct a ^{max.} heap tree for the given elements:

19, 14, 22, 21, 35, 18, 16, 15, 42, 23

Solⁿ:

(19)

⇒

(19)
14

⇒

(19)
14 22

Heap tree property violated ⇒ Shift up applied

(22)
21 19
14 35

Shift up Process

(22)
14 19
21

Shift up process

(22)
35 19
14 21

Shift up process

(35)
22 19
14 21

(35)
22 19
14 21 18

⇓

(35)
22 19
15 21 18 16
14 42

Shift up process

(35)
22 19
14 21 18 16
15

Shift up process

(35)
22 19
42 21 18 16
14 15

Shift up process

(35)
42 19
22 21 18 16
14 15

Shift up process

(42)
35 19
22 23 18 16
14 15 21

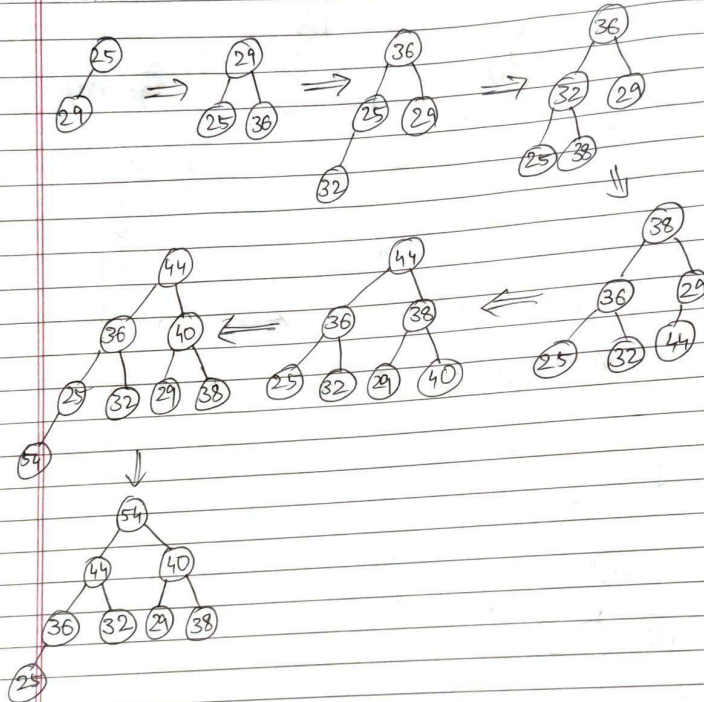
Shift up process

(42)
35 19
22 21 18 16
14 15 23

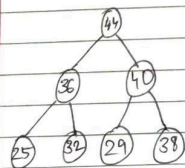
2 -

Q. 25, 29, 36, 32, 38, 44, 40, 54

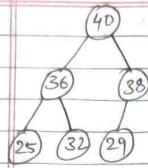
Soln:	Max:
-------	------



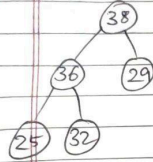
Sorting :



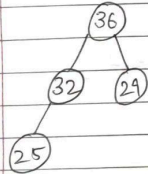
$$x = \{44, 36, 40, 25, 32, 29, 38, 54\}$$



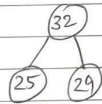
$$x = \{ 40, 36, 38, 25, 32, \cancel{29}, 44, 54 \}$$



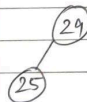
$$x = \{ 38, 36, 29, 25, 32, \cancel{40}, \cancel{44}, 54 \}$$



$$x = \{ 36, 32, 29, 25, 38, 40, 44, 54 \}$$



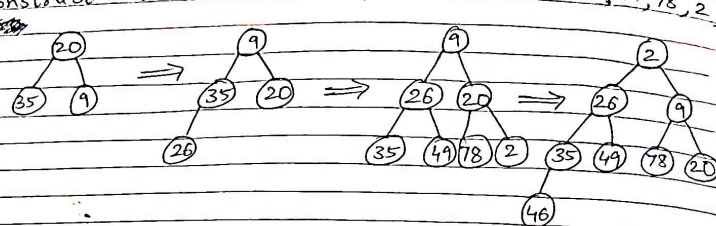
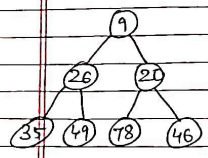
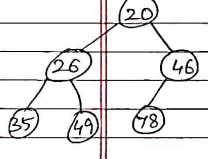
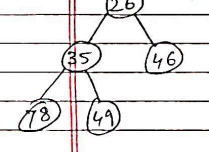
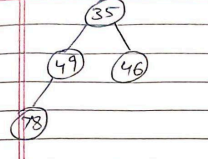
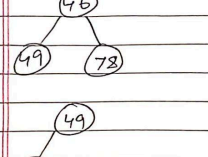
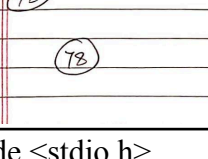
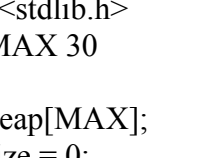
$$x = \{ 32, 25, 29, 36, 38, 40, 44, 54 \}$$



$$x = \{ 29, 25, 32, 36, 38, 40, 44, 54 \}$$



$$x = \{ \cancel{25}, 29, 32, 36, 38, 40, 44, 54 \}$$

	<p>Q. Construct & sort min. heap : 20, 35, 9, 26, 49, 78, 2, 46</p> <p>Sol.:</p>  <p>Sorting :</p>  <p>$x = \{ 9, 26, 20, 35, 49, 78, 46, 2 \}$</p>  <p>$x = \{ 20, 26, 46, 35, 49, 78, 9, 2 \}$</p>  <p>$x = \{ 26, 35, 46, 78, 49, 20, 9, 2 \}$</p>  <p>$x = \{ 35, 49, 46, 78, 26, 20, 9, 2 \}$</p>  <p>$x = \{ 46, 49, 78, 35, 26, 20, 9, 2 \}$</p>  <p>$x = \{ 49, 78, 46, 35, 26, 20, 9, 2 \}$</p>  <p>$x = \{ 78, 49, 46, 35, 26, 20, 9, 2 \}$</p>
<p>Program(Code)</p>	<pre> #include <stdio.h> #include <stdlib.h> #define MAX 30 int maxHeap[MAX]; int maxSize = 0; int minHeap[MAX]; int minSize = 0; void swap(int *p, int *q){ int temp = *p; *p = *q; *q = temp; </pre>

```

}

void maxHeapify(int i){
    int left = 2*i;
    int right = (2*i)+1;
    int s=i;

    if(left<=maxSize && maxHeap[left]>maxHeap[s]){
        s = left;
    }

    if(right<=maxSize && maxHeap[right]>maxHeap[s]){
        s = right;
    }

    if(s!=i){
        swap(&maxHeap[s], &maxHeap[i]);
        maxHeapify(s);
    }
}

void minHeapify(int i){
    int left = 2*i;
    int right = (2*i)+1;
    int s=i;

    if(left<=minSize && minHeap[left]<minHeap[s]){
        s = left;
    }

    if(right<=minSize && minHeap[right]<minHeap[s]){
        s = right;
    }

    if(s!=i){
        swap(&minHeap[s], &minHeap[i]);
        minHeapify(s);
    }
}

void insertMax(int val){
    if(maxSize>=MAX-1){
        printf("Max size reached...can't add\n");
        return;
    }
    maxHeap[++maxSize] = val;

    int i = maxSize;
    int parent = i/2;
    while(i>1 && maxHeap[i]>maxHeap[parent]){

```

```

        swap(&maxHeap[i], &maxHeap[parent]);
        i = parent;
        parent = i/2;
    }
}

void insertMin(int val){
    if(minSize>=MAX-1){
        printf("Max size reached...can't add\n");
        return;
    }
    minHeap[++minSize] = val;

    int i = minSize;
    int parent = i/2;
    while(i>1 && minHeap[i]<minHeap[parent]){
        swap(&minHeap[i], &minHeap[parent]);
        i = parent;
        parent = i/2;
    }
}

int extractMax(){
    if(maxSize==0){
        printf("Heap is empty...can't extract max !\n");
        return -1;
    }
    int max = maxHeap[1];
    maxHeap[1] = maxHeap[maxSize--];
    maxHeapify(1);
    return max;
}

int extractMin(){
    if(minSize==0){
        printf("Heap is empty...can't extract min !\n");
        return -1;
    }
    int min = minHeap[1];
    minHeap[1] = minHeap[minSize--];
    minHeapify(1);
    return min;
}

void maxHeapSort(){
    int temp = maxSize;
    while(maxSize>1){
        swap(&maxHeap[1], &maxHeap[maxSize--]);
        maxHeapify(1);
    }
}

```

```

    maxSize = temp;
}

void minHeapSort(){
    int temp = minSize;
    while(minSize>1){
        swap(&minHeap[1], &minHeap[minSize--]);
        minHeapify(1);
    }
    minSize = temp;
}

void printMaxHeap(){
    for(int i=1; i<=maxSize; i++){
        printf("%d ", maxHeap[i]);
    }
    printf("\n");
}

void printMinHeap(){
    for(int i=1; i<=minSize; i++){
        printf("%d ", minHeap[i]);
    }
    printf("\n");
}

int main(){
    int t = 1, o, temp;
    do{
        printf("Choose an operation to perform - 1. Insert    2. Extract    3.
Heap Sort    4. Print    5. Exit\nEnter your choice : ");
        scanf("%d", &o);
        switch(o){
            case 1 :
                printf("Enter the value to be inserted : ");
                scanf("%d", &temp);
                insertMax(temp);
                insertMin(temp);
                break;

            case 2 :
                temp = extractMax();
                printf("Extracted max : %d\n", temp);
                temp = extractMin();
                printf("Extracted min : %d\n", temp);
                break;

            case 3 :
                maxHeapSort();
                printf("Max Heap after sorting : ");

```


	<pre>printMaxHeap(); minHeapSort(); printf("Min Heap after sorting : "); printMinHeap(); break; case 4 : printf("Max Heap : "); printMaxHeap(); printf("Min Heap : "); printMinHeap(); break; case 5 : printf("Exiting..."); t=0; break; default : printf("Enter a valid input !\n"); } } while(t); return 0; }</pre>
--	--

Output

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter your choice : 1
Enter the value to be inserted : 19
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 14
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 22
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 21
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 35
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 18
Enter a valid input !
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 18
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 16
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 15
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 42
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 1
Enter the value to be inserted : 23
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 4
Max Heap : 42 35 19 22 23 18 16 14 15 21
Min Heap : 14 15 16 19 23 22 18 21 42 35
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 42
Extracted min : 14
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 35
Extracted min : 15
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 4
Max Heap : 23 22 19 15 21 18 16 14
Min Heap : 16 19 18 21 23 22 42 35
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter your choice : 2
Extracted max : 42
Extracted min : 14
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 35
Extracted min : 15
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 4
Max Heap : 23 22 19 15 21 18 16 14
Min Heap : 16 19 18 21 23 22 42 35
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 3
Max Heap after sorting : 14 15 16 18 19 21 22 23
Min Heap after sorting : 42 35 23 22 21 19 18 16
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 14
Extracted min : 42
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 23
Extracted min : 16
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 4
Max Heap : 22 15 16 18 19 21
Min Heap : 18 35 23 22 21 19
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 22
Extracted min : 18
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 2
Extracted max : 21
Extracted min : 19
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 4
Max Heap : 19 15 16 18
Min Heap : 21 35 23 22
Choose an operation to perform - 1. Insert    2. Extract    3. Heap Sort    4. Print    5. Exit
Enter your choice : 5
Exiting...
```

○ PS C:\Mahadev\S.E\DS\Lab Sessions> █

Conclusion	<p>In this experiment, we successfully implemented essential operations for managing a max-heap structure, including insertion, deletion, and heap property maintenance through heapifyUp and heapifyDown. The insertMax function efficiently adds new elements while preserving the max-heap order, and the delete operation ensures that the heap is reorganized after removing the root element. These operations demonstrate the utility of heaps for efficient priority queue management, ensuring constant access to the maximum element and efficient reordering upon updates.</p>
-------------------	---