

Data Structures and Algorithms

Assignment - IV

1 Implement the following operations for a min-heap:

a. Insert a new element into the min-heap.

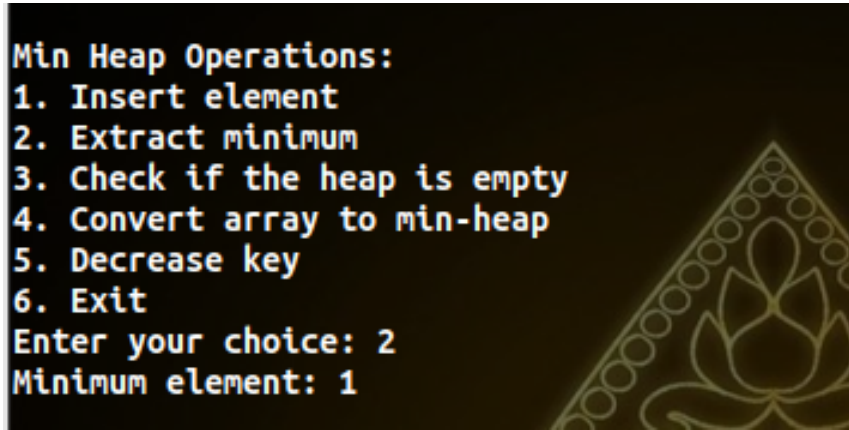
```
TURING
File Edit View Terminal Tabs Help
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/
min
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/
Enter the initial capacity of the MinHeap: 3

Min Heap Operations:
1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit
Enter your choice: 1
Enter element to insert: 1
Element 1 inserted.

Min Heap Operations:
1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit
Enter your choice: 1
Enter element to insert: 2
Element 2 inserted.

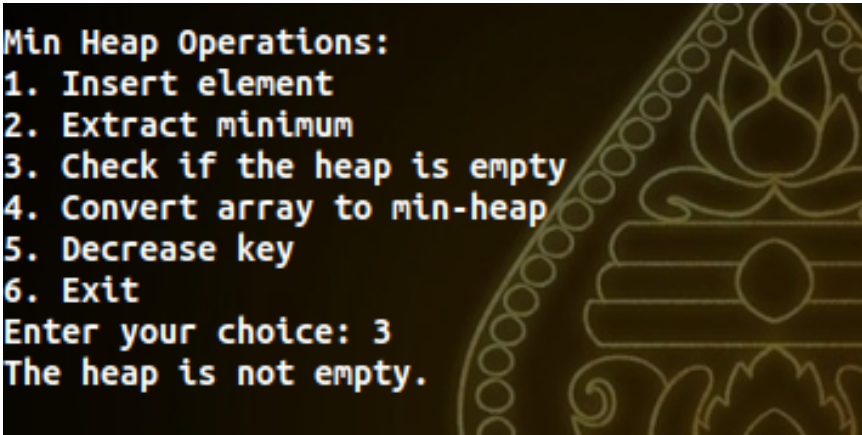
Min Heap Operations:
1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit
Enter your choice: 1
Enter element to insert: 33
Element 33 inserted.
```

b. Extract the minimum element from the min-heap.



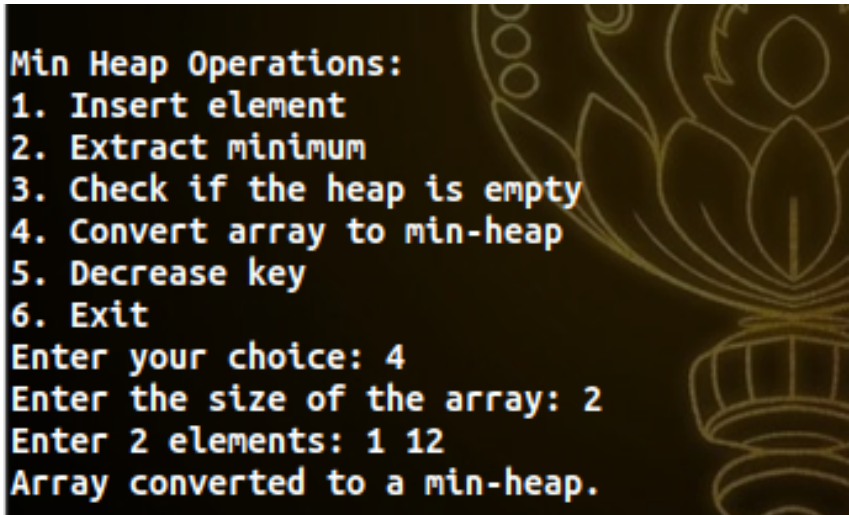
```
Min Heap Operations:
1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit
Enter your choice: 2
Minimum element: 1
```

c. Check if the min-heap is empty.



```
Min Heap Operations:
1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit
Enter your choice: 3
The heap is not empty.
```

d. Convert an array of elements into a min-heap.



```
Min Heap Operations:
1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit
Enter your choice: 4
Enter the size of the array: 2
Enter 2 elements: 1 12
Array converted to a min-heap.
```

e. Decrease Key - decrease the value of a specific element in the min-heap and maintain the heap property.

Min Heap Operations:

1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit

Enter your choice: 5

Enter the index and new value (space-separated)

Min Heap Operations:


1. Insert element
2. Extract minimum
3. Check if the heap is empty
4. Convert array to min-heap
5. Decrease key
6. Exit

Enter your choice: 6

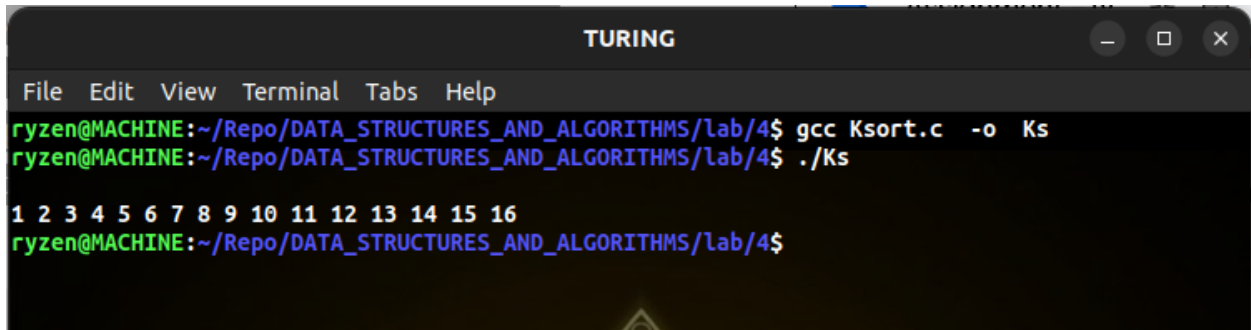
Program terminated.

2. implement a priority queue using max-heap.

```
TURING
File Edit View Terminal Tabs Help
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$ gcc PriorityQueue.c -o pQ
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$ ./PQ
bash: ./PQ: No such file or directory
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$ ./pQ
Choose an operation:
1. Insert
2. Delete Max
3. Peek Max
4. Exit
1
Enter key to insert: 1
Choose an operation:
1. Insert
2. Delete Max
3. Peek Max
4. Exit
1
Enter key to insert: 2
Choose an operation:
1. Insert
2. Delete Max
3. Peek Max
4. Exit
1
Enter key to insert: 23
Choose an operation:
1. Insert
2. Delete Max
3. Peek Max
4. Exit
2
Choose an operation:
1. Insert
2. Delete Max
3. Peek Max
4. Exit
3
Max priority element: 2
Choose an operation:
1. Insert
2. Delete Max
3. Peek Max
4. Exit
4
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$
```



3. Implement a function that merges k sorted arrays into a single sorted array using a min-heap.



```
TURING
File Edit View Terminal Tabs Help
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$ gcc Ksort.c -o Ks
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$ ./Ks
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ryzen@MACHINE:~/Repo/DATA_STRUCTURES_AND_ALGORITHMS/lab/4$
```