Priority Queue Implementation using Linked list data structure

```
PUSH(HEAD, DATA, PRIORITY):
    • Step 1: Create new node with DATA and PRIORITY
    • Step 2: Check if HEAD has lower priority. If true follow Steps 3-4 and end. Else goto Step 5.
    • Step 3: NEW -> NEXT = HEAD
    • Step 4: HEAD = NEW
    • Step 5: Set TEMP to head of the list
    • Step 6: While TEMP -> NEXT != NULL and TEMP -> NEXT -> PRIORITY > PRIORITY
    • Step 7: TEMP = TEMP -> NEXT
      [END OF LOOP]
    • Step 8: NEW -> NEXT = TEMP -> NEXT
    • Step 9: TEMP -> NEXT = NEW
    • Step 10: End
```

```
POP(HEAD):
    • Step 1: Set the head of the list to the next node in the list. HEAD = HEAD -> NEXT.
    • Step 2: Free the node at the head of the list
    • Step 3: End
```

```
PEEK(HEAD):
    • Step 1: Return HEAD -> DATA
    • Step 2: End
```

Priority Queue Implementation using HEAP data structure

Step1: Create a function heapify() to heapify the elements in the Binary Tree if any changes are made.

Step2: Find the largest among root, left child, and right child, then recursively call the same function until the root element matches the largest element.

Step3: Create a function insert() to insert an element into the tree, which takes an array and the number which is to be inserted as input.

Step 4: If the size of the array is zero, then this number will be the root, else append the number and call the heapify function recursively to heapify the elements.

Step5: Create a function deleteNode() that deletes the selected element from the tree

Step 6: Delete the element and again heapify the elements recursively.

Step 7: Insert elements into an empty array using the insert() function then try deleting an element from the tree.