14/06/2021

Name	Branch	Section	Roll	Reg No	[illegible]
AYUSH GOYAL	CSE	D	62	1AGAC5522	[illegible]

DAA LAB ENDSEM [Signature... [illegible] 14/6/2021]

(Ans:)

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
        int data;
        int priority;
  } NODE;
  void heapify (NODE arr[], int n) {
        for (int i = n/2 ; i >= 1; i--) {
              NODE v = arr[i];
              int k = i;
              int flag = 0;
              while (flag == 0  && (2 * k) <= n) {
                    int j = 2 * k;
                    if (j < n) {
                          if (arr[j].priority < arr[j+1].priority)
                          {
                             j++;
                          }
                    }
                    if (v.priority < arr[j].priority) {
                          arr[k] = arr[j];
                           k = j;
                    }
                    else {
                          flag = 1;
                    }
              }
              arr[k] = v;
        }
        return;
  }
```

```
int deleteMax (NODE arr[], int n) {

        NODE x = arr[1];
        arr[1] = arr[n];
        arr[n] = x;
        n = n-1;            // we use root deletion method
        heapify (arr, n);   // We heapify after swapping root
        return n;
    }

  void heapifyRoot (NODE arr[], int n) {   // We don't heapify entire data
                                             only ancestors in this function
        if (n <= 1)   return;
        if (arr[n].priority > arr[n/2].priority) {

                NODE x = arr[n];
                arr[n] = arr[n/2];
                arr[n/2] = x;
           }
        heapifyRoot (arr, n/2); //We only check for ancestors
    }

  int insert (NODE arr[], int n, NODE cur) {

        n = n+1;
        arr[n] = cur;
        heapifyRoot (arr, n);
        return n;
    }

   int deleteAny (NODE arr[], int n, int del) {

        int p = -1; //position index
        for (int i = 1; i <= n; i++) {
              if (arr[i].data == del) {
                      p = i;
                      break;
              }
        }
        if (p == -1) {
        &    printf("\n Element not present");
             return n;
        }
```

Scanned by CamScanner

```
~~for~~ · for(int i=p; i<n; i++) {
              arr[i]= arr[i+1];
        }
        return (n-1);

  }

int main(){
      NODE arr[500];
      NODE val;
      int n=0, temp, ch;

      do{
        printf("\n1. Insert    2. Display   3. Delete max priority
                  element    4. Delete element by value    5. Exit
                  Enter choice :   ");

          scanf("%d", &ch);
          switch(ch){
                case 1:  printf("Enter element :");
                         scanf("%d", & val.data);
                         printf("Enter priority: ");
                         scanf("%d", & val.priority);

          /* In our case, the value of data and priority will
               be same, ∴ we can take priority as value
               of data itself. This is because given in question,
               increasing order of value)*/
                            n= insert(arr, n, val);
                            break;

                case 2 :   printf("Displaying PQ:");
                           for(int i=1; i<=n; i++){
                               ~~print~~
                               printf("%d", arr[i].data);
                           }
                           break;

                case 3 :   ~~if(arr==0)~~ if(n==0){
                                printf("Queue is Empty");
                           }
                           else{
                              n= deletemax(arr,n);
                           printf("Element deleted is : %d
                                  with priority: %d\n",
                                arr[n+1].data, arr[n+1].priority);
                           }
                        break;
```

```
                    case 4:  printf ("Enter which element :");
                             scanf("%d", &temp);
                             n= deleteAny (arr, n, temp);
                             break;
                    case 5:  exit(1);
                             break;
                    default: break;
            }
        } while (ch! =5);
    return 0;
}
```