PRIORITY QUEUES

A priority Queues is a data structure in which each element is assigned a priority. The priority of the element will be used to determine the order in which the element will be processed.

→ An element with higher priority is processed before an element with a lowes priority.

→ Two element with the same priority are processed on a FCFS.

Priority Queues are widely used in an operating system to execute the highest priority process first. The priority of the process may be set based on the CPU time it require to get executed completely.

CPU time is not the only factor that determines the priority, rather it is just one among many several factors.
Another factor is the importance of one process over the others.

IMPLEMENTATION OF PRIORITY QUEUES.

There are two ways to implement Priority Queues.→

We can either use a ~~stored~~ sorted list to store the element so that when an element has to be taken out, the queue will not have to be searched for the element with the highest priority or we can use an unsorted list so that insertion are always done at the end of the list.

Every time when an element has to be removed from the list, the element with the highest priority will be searched and removed.

While a sorted list takes O(n) time to insert an element in the list, it takes only O(1) time to delete an element.

On the contrary, an unsorted list will take O(1) time to insert an element and O(n) time to delete an element from the list.

Practically both these techniques are inefficient and usually a blend of these two approaches is adopted and roughly O(log n) time or less.

Pseudo Code:-

```
struct node
{
  int data;
  int priority;
  struct node *next;
}

struct node *start = NULL;
struct node * insert (struct node *);
struct node * delete (struct node *);
void display (struct node *);

  int main()
  {
    =
  }
```

```
struct node *insert( struct node * start)
  {
    int val, pri;
    struct node *ptr, *p;
    ptr = (struct node*) malloc(size of (struct node));
    printf ("\n Enter the value & its priority: ");
     &canf("%d %d" , &val, &pri);
     ptr->data =val;
     ptr-> priority = pri;
     if (start == NULL || pri < start -> priority )
           {
             ptr->next = start;
              start = ptr;
           }
      else
          {
           p = start;
           while (p-> next != NULL && p->next-> priority <= pri)
                  {  p =p-> next;
                  }
                ptr->next = p->next;
                  p-> next = ptr;
                }
           return start;
      }

struct node *delete( struct node * start)
    { struct node * ptr
      if (start == NULL)
          { printf(" Underflow");
            return;
          }
      else
```

```
    {
      ptr = start;
      printf("\n Deleted item is : %d", ptr->data);
        start = start -> next;
        free(ptr);
    }
    return start;
}

void display (struct node * start)
    {
    struct node * ptr;
      ptr = start;
    if (start == NULL)
        printf ("\n Queues is empty");

    else
     { printf ("\n PRIORITY QUEUE is ");

      while (ptr != NULL)
        {
        printf(" \t %d [priority = %d]", ptr->data,
                                          ptr->priority);

          ptr = ptr -> next;

        }.
```