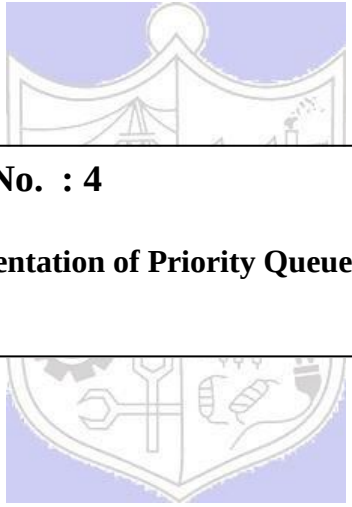


Experiment No. : 4

Title: Implementation of Priority Queue

Batch: B1 **Roll No.: 16010420133** **Experiment No.: 4**

Aim: Write a menu driven program to implement a static priority queue using supporting following operations.

1. Create empty queue,
2. Insert an element on the queue,
3. Delete an element from the queue,
4. Display front element
5. Display all elements of the queue.

Resources Used: Turbo C/ C++/JAVA editor and compiler.

Theory:

Queue -A queue is an ordered list in which insertion and deletion happens two different ends.The insertion happens from the *rear* and the deletion takes place at the *front*. It works with the FIFO concept i.e. first in first out. Basic operations of Queue are enqueue, dequeue, isempty, etc.

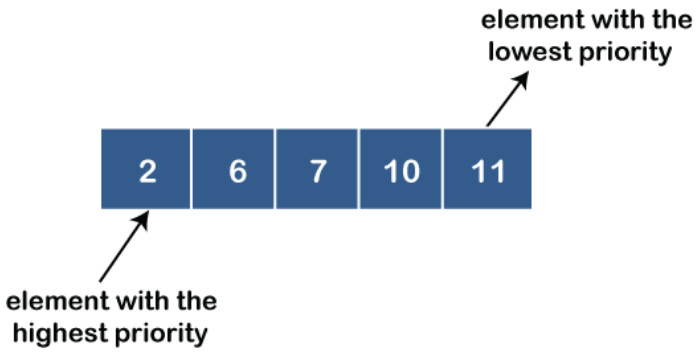
Priority Queue : A priority queue is an abstract data type that behaves similarly to the normal queue except that each element has some priority, i.e., the element with the highest priority would come first in a priority queue. The priority of the elements in a priority queue will determine the order in which elements are removed from the priority queue. The priority queue supports only comparable elements, which means that the elements are either arranged in an ascending or descending order.

- Every element in a priority queue has some priority associated with it.
- An element with the higher priority will be deleted before the deletion of the lesser priority.
- If two elements in a priority queue have the same priority, they will be arranged using the FIFO principle.

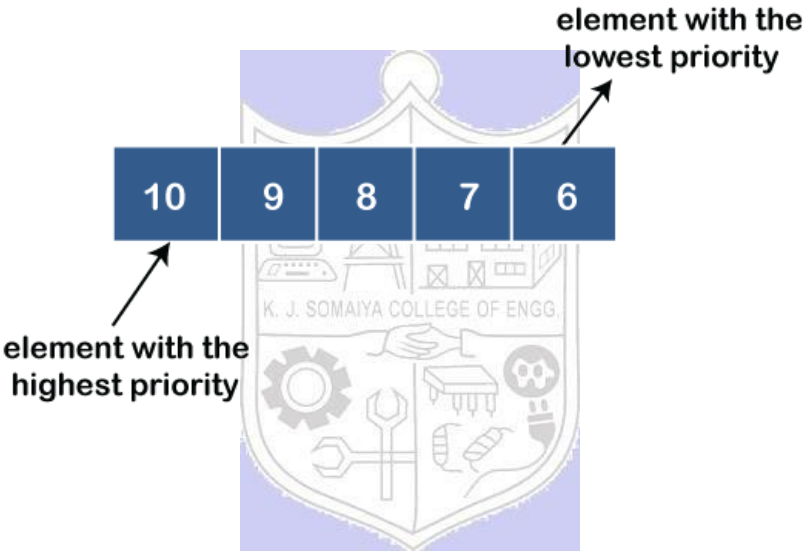
Types of Priority Queue -

There are two types of priority queue:

Ascending order priority queue: In ascending order priority queue, a lower priority number is given as a higher priority in a priority. For example, we take the numbers from 1 to 5 arranged in an ascending order like 1,2,3,4,5; therefore, the smallest number, i.e., 1 is given as the highest priority in a priority queue.



Descending order priority queue: In descending order priority queue, a higher priority number is given as a higher priority in a priority. For example, we take the numbers from 1 to 5 arranged in descending order like 5, 4, 3, 2, 1; therefore, the largest number, i.e., 5 is given as the highest priority in a priority queue.



Algorithm :

1. **typedef createQueue():** This method creates and empty queue by setting initial values of front=0, rear=-1.
2. **void enqueue(typedef val) :**

This operation adds an item at the rear of the queue. Before insert operation, ensure that there is a room for the new item. If there is not enough room, then the queue is in an 'Overflow' state.

3. **typedef dequeueMax() :** This operation removes the item at the *front* of the *Queue* which has maximum priority and returns it to the user. Because the front item has been removed, the next item becomes the front. When the last item in the queue is deleted, it must be set to its empty state. If *dequeue* is called when the queue is empty, then it's in an 'Underflow' state.
4. **typedef getFront() :** This operation will return maximum priority item of the queue at that instance.
5. **void displayAll():** This operation will display all item present in the queue at that instance.
6. **boolean isEmpty() :** This operation will check whether *Queue* is empty or not at a given instance. The function will return

   0 if *Queue* is not empty.
   1 if *Queue* is empty.

8. **boolean isFull() :** This operation will check whether *Queue* is full or not at a given instance. The function will return

   0 if *Queue* is not full.
   1 if *Queue* is full.

9. **int size() :** This operation will count the total number of elements present in the *Queue* at a given instance and return the count.

   0 if Queue is empty.
   n if Queue is having n no. of elements.

Activity: students are expected to implement priority queue using **array.**

NOTE : All functions should be able to handle boundary(exceptional) conditions.

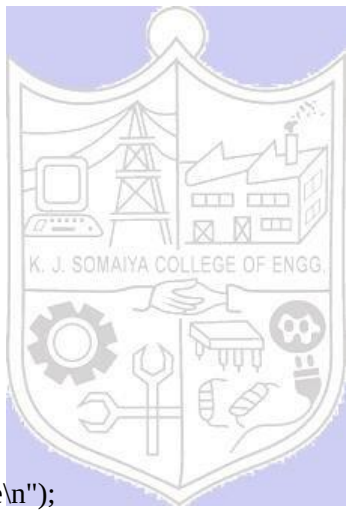
Results: A program implementing solution depicting the correct behaviour of priority queue and capable of handling all possible exceptional conditions and the same is reflecting clearly in the output.

Code:

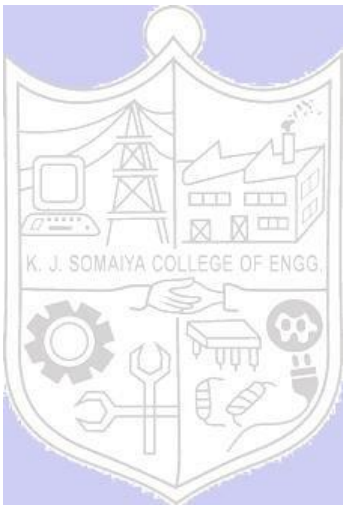
```
#include<stdio.h>

#include <stdbool.h>

#include<stdlib.h>
```

```
int queue_arr[MAX];
int rear=-1;
int front=0;
typedef createQueue();
void enqueue(typedef val);
int Delete();
typedef getFront(+);
void displayAll();
bool isFull();
bool isEmpty();



int main()
{
    int choice,val;
    while(1)
    {
        printf("\n1.Create Queue \n2.Insert \n3.Delete \n4.Display element at the front \n5.Display all elements of the queue \n6.Quit \n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);


        switch(choice)
        {
        case 1: createQueue();
            break;


        case 2:
```

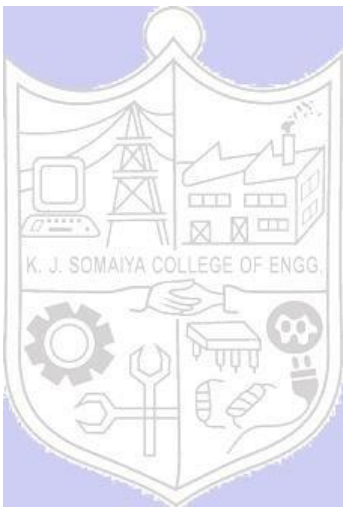
```
            printf("\nInput Element for Addition in Queue: ");
            scanf("%d",&val);
            enqueue(val);
            break;
        case 3:
            val=Delete();
            printf("\nDeleted Element: %d\n",val);
            break;
        case 4:
            printf("\nFirst Element: %d\n",getFront());
            break;
        case 5:
            displayAll();
            break;
        case 6:
            exit(1);
        default:
            printf("\nWrong choice\n");
        }
    }

    return 0;

}

typedef createQueue() {
  front=0;
```



```
  rear = -1;
}

void enqueue(typedef val)
{
      if( isFull() )
      {
            printf("\nQueue Overflow\n");
            return;
      }
      rear=rear+1;
      queue_arr[rear]=val ;
}

int Delete()
{
      int val;
      if( isEmpty() )
      {
            printf("\nQueue Underflow\n");
            exit(1);
      }
      val=queue_arr[front];
      front=front+1;
      return val;
}
```

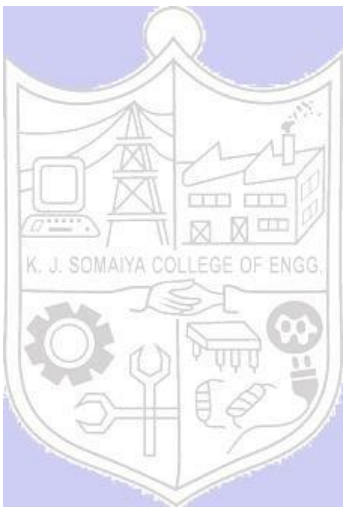


```
typedef getFront()
{
    if( isEmpty() )
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    return queue_arr[front];
}

bool isEmpty()
{
    if( front==-1 || front==rear+1 )
        return 1;
    else
        return 0;
}

bool isFull()
{
    if( rear==MAX-1 )
        return 1;
    else
        return 0;
}

void displayAll()
```



```
{
    int i;
    if ( isEmpty() )
    {
        printf("\nQueue is empty\n");
        return;
    }
    printf("\nQueue is :\n\n");
    for(i=front;i<=rear;i++)
        printf("%d  ",queue_arr[i]);
}
```

Output:



```
1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 1

1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 2

Input the element for adding in queue : 4

1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 2

Input the element for adding in queue : 5

1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 4

Element at the front is 45

1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit
```

```
Enter your choice : 5

Queue is :

45  78
1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 3

Deleted element is  45

1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 5

Queue is :

78
1.Create Queue
2.Insert
3.Delete
4.Display element at the front
5.Display all elements of the queue
6.Quit

Enter your choice : 6

Process returned 1 (0x1)   execution time : 55.038 s
Press any key to continue.
```

Outcome:

Apply linear and non-linear data structure in application development.

Conclusion:

A menu driven program to implement a static priority queue was implemented and the concepts of queue and priority queue were understood

Grade: AA / AB / BB / BC / CC / CD /DD:

Signature of faculty in-charge with date :

References:

Books/ Journals/ Websites:

- Y. Langsam, M. Augenstin and A. Tannenbaum, "Data Structures using C", Pearson Education Asia, 1st Edition, 2002.