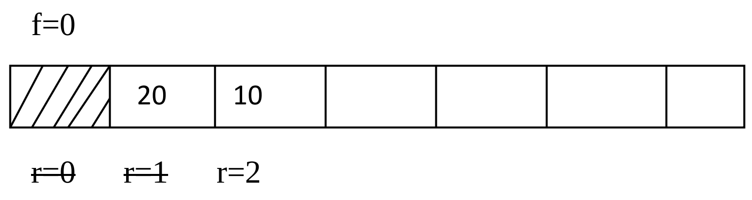
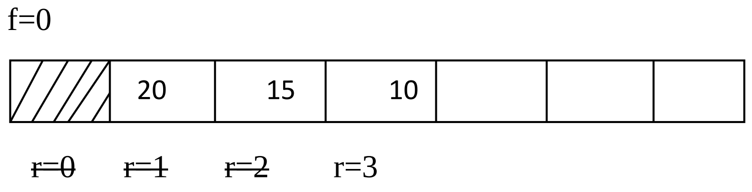
<u>Priority Queue</u>

A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue. The illustration of priority queue is as follows:

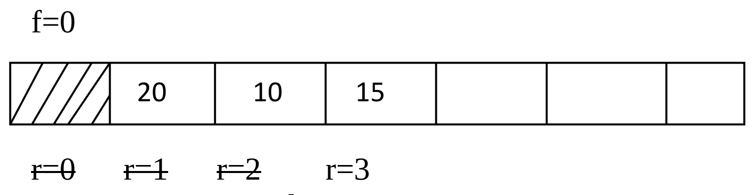


item =15



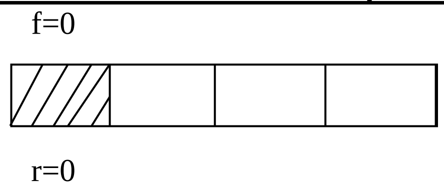
Priority queue

item =15

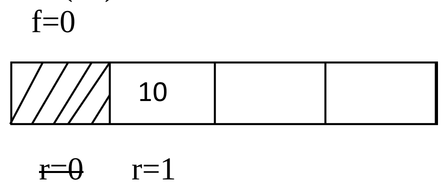


Normal queue

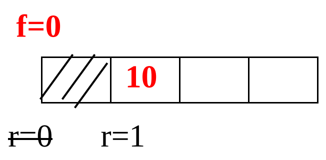
<u>The insertion and deletion operations on priority queue are as follow:</u>



insert (10)

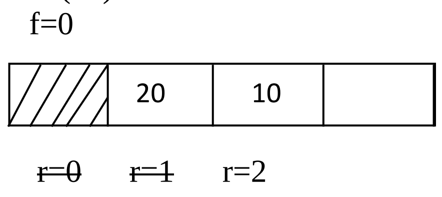


Normal queue

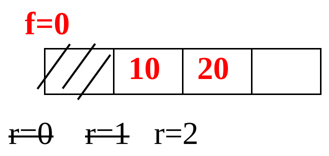


insert (20)

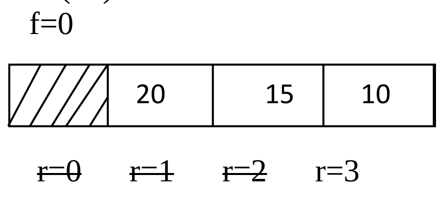
Note: We consider high integer = high priority



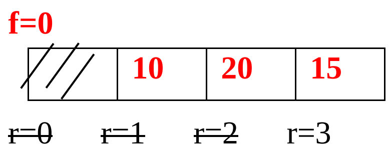
Normal queue



insert (15)



Normal queue

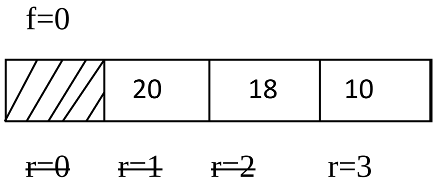


<u>Algorithm for insertion into priority queue</u>

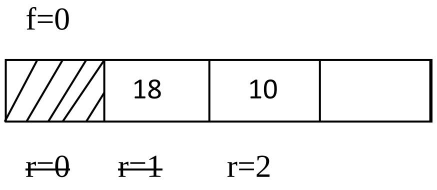
```
void insert_by_priority (int data){
if (rear >= MAX - 1)    {
        printf("\nQueue overflow no more elements can be inserted");
        return;
   }
   if ((front == -1) && (rear == -1))   {
        rear++;
        pri_que[rear] = data;
        return;
   }
   else {
        for (i = 0; i <= rear; i++)
            {
               if (data >= pri_que[i])
                  {
                   for ( j = rear + 1; j > i; j--)
                     {
                       pri_que[j] = pri_que[j - 1];
                     }
                  pri_que[i] = data;
                  rear++;
                  return;
                }
             }
         pri_que[i] = data;
         rear++;
       }
   }
```

Deletion:

delete(20)



delete(20)



Algorithm for deletion form priority queue

```
void delete_by_priority (int data) {
    if ((front= = -1) && (rear = = -1))
      {
        printf("\nQueue is empty no elements to delete");
        return;
       }
        for (i = 0; i <= rear; i++)
          {
             if (data == pri_que[i])
               {
                 for (; i < rear; i++)
                   {
                     pri_que[i] = pri_que[i + 1];
                   }
                pri_que[i] = -99;
                rear--;
                if (rear == -1)
                front = -1;
                return;
             }
         }
     printf("\n%d not found in queue to delete", data);
  }
```

Display Algorithm of priority queue:

```
void display_pqueue( ){
            front = 0;
             if ((front == -1) && (rear == -1))    {
               printf("\nQueue is empty");
              return;
             }
         for (; front <= rear; front++){
          printf(" %d ", pri_que[front]);
         }
    }
```

Write a Program in C for priority queue insertion and deletion:

```
 /* C Program to Implement Priority Queue to insert and delete eElements */
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
void insert_by_priority(int);
void delete_by_priority(int);
void display_pqueue();
int pri_que[MAX];
int front, rear;
void main()
{
      int n, ch;
    printf("\n1 - Insert an element into queue");
    printf("\n2 - Delete an element from queue");
    printf("\n3 - Display queue elements");
    printf("\n4 - Exit");
             front = rear = -1;
    while (1)
    {
        printf("\nEnter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("\nEnter value to be inserted : ");
            scanf("%d",&n);
            insert_by_priority(n);
            break;
        case 2:
            printf("\nEnter value to delete : ");
            scanf("%d",&n);
            delete_by_priority(n);
            break;
        case 3:
            printf("\nDisplay Values as Priority:= ");
            display_pqueue();
            break;
            case 4:
            exit(0);
            default:
            printf("\nChoice is incorrect, Enter a correct choice");
        }
    }
}
/* Function to insert value into priority queue */
void insert_by_priority(int data)
{
             if (rear >= MAX - 1)
    {
        printf("\nQueue overflow no more elements can be inserted");
        return;
    }
    if ((front == -1) && (rear == -1))
    {
         rear++;
        pri_que[rear] = data;
        return;
    }
    else
             {
             for (int i = 0; i <= rear; i++)
             {
        if (data >= pri_que[i])
        {
            for (int j = rear + 1; j > i; j--)
            {
                pri_que[j] = pri_que[j - 1];
            }
            pri_que[i] = data;
              rear++;
            return;
        }
             }
           pri_que[i] = data;
             rear++;
             }
}
/* Function to delete an element from queue */
void delete_by_priority(int data)
{
    int i;
    if ((front==-1) && (rear==-1))
    {
        printf("\nQueue is empty no elements to delete");
        return;
    }
    for (i = 0; i <= rear; i++)
    {
        if (data == pri_que[i])
        {
            for (; i < rear; i++)
            {
                pri_que[i] = pri_que[i + 1];
            }
        pri_que[i] = -99;
        rear--;
        if (rear == -1)
            front = -1;
        return;
        }
    }
    printf("\n%d not found in queue to delete", data);
}
/* Function to display queue elements */
void display_pqueue()
{
    front = 0;
             if ((front == -1) && (rear == -1))
              {
        printf("\nQueue is empty");
        return;
    }
    for (; front <= rear; front++)
    {
        printf(" %d ", pri_que[front]);
    }
}
```

Output on Monitor:

```
1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 1
Enter value to be inserted : 5
Enter your choice : 1
Enter value to be inserted : 9
Enter your choice : 1
Enter value to be inserted : 7
Enter your choice : 1
Enter value to be inserted : 3
Enter your choice : 1
Enter value to be inserted : 5
Enter your choice : 1
Enter value to be inserted : 6
Queue overflow no more elements can be inserted
Enter your choice : 3
Display Values as Priority:=  9  7  5  5  3
Enter your choice : 2
Enter value to delete : 5
Enter your choice : 3
Display Values as Priority:=  9  7  5  3
Enter your choice : 2
Enter value to delete : 9
Enter your choice : 3
Display Values as Priority:=  7  5  3
Enter your choice : 4
Press any key to continue
```

Application of Priority Queue:

- Shortest path Disjktras's algorithm
- Prim's algorithm of minimum spanning tree
- Data compression techniques like Huffman coding
- Heapsort (for insertion & deletion)
- In operating system such as priority scheduling, load balancing and interrupt handling.