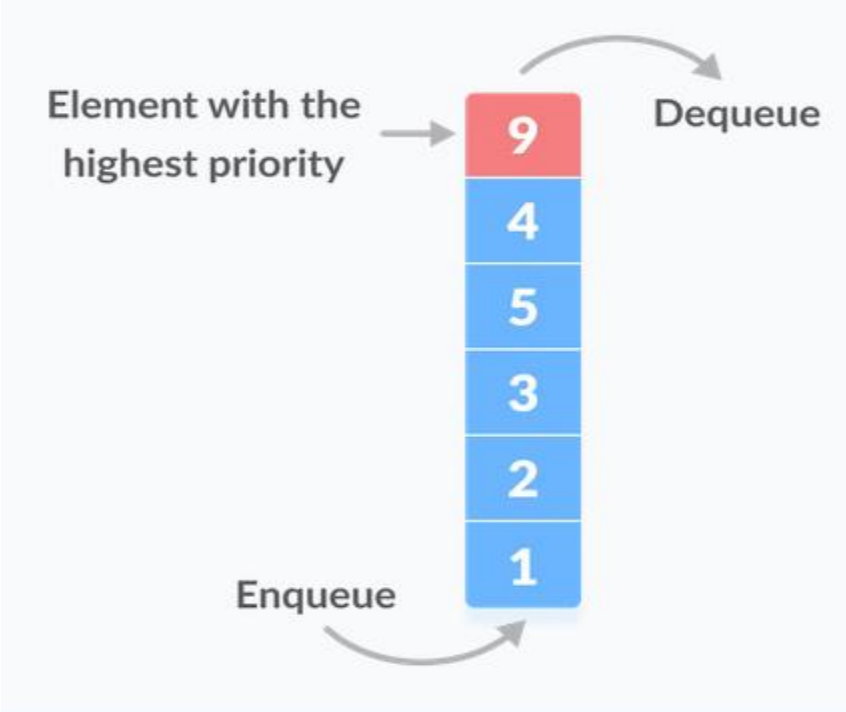
Priority Queue

A **priority** queue is **a special type of queue in which each element is associated with a priority and is served according to its priority**. If elements with the same priority occur, they are served according to their order in the queue.

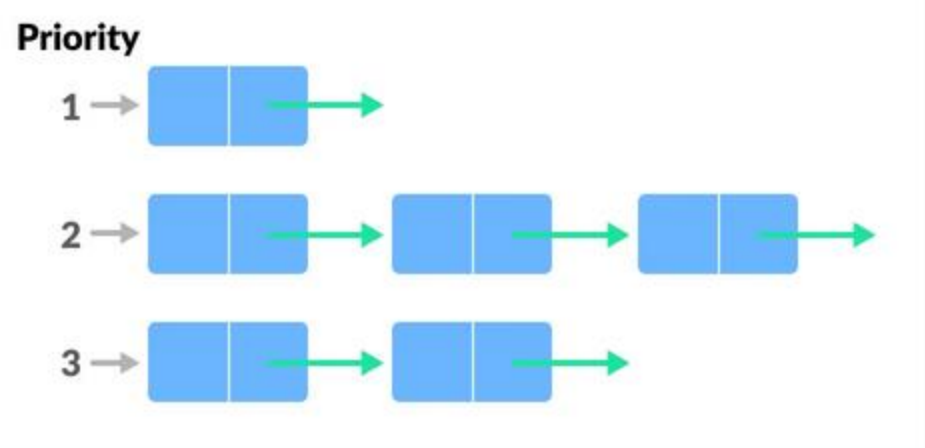
For example, The element with the highest value is considered as the highest priority element. However, in other cases, we can assume the element with the lowest value as the highest priority element. In other cases, we can set priorities according to our needs.



Removing Highest Priority Element

Difference between Priority Queue and Normal Queue

In a queue, the **first-in-first-out rule** is implemented whereas, in a **priority queue**, the values are removed **on the basis of priority**. The element with the highest priority is removed first.



Priority Queue Representation

Example :

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

void insert(int);
void del();
void display();
int *pq1,*pq2,*pq3,fr1=-1,fr2=-1,fr3=-1,r1=-1,r2=-1,r3=-1,p,n,i;

void main()
{
      int ch,item;
        clrscr();
        printf("\n Enter size:");
        scanf("%d",&n);
        pq1=(int*)malloc(n*sizeof(int));
        pq2=(int*)malloc(n*sizeof(int));
        pq3=(int*)malloc(n*sizeof(int));
   while (1)
   {
        printf("\n1:Insert \n2.Delete \n3.Display \n4.Exit \n");
```

```
    printf("\nEnter choice :");
    scanf("%d", &ch);

    switch (ch)
    {
    case 1:
                if(r1!=(n-1)||r2!=(n-1)||r3!=(n-1))
                {
                        printf("Enter queue element:");
                        scanf("%d",&item);
                        printf("\n Enter priority(1,2,3):");
                        scanf("%d",&p);

                }
                if((p==1)||(p==2)||(p==3))
                        insert(item);
                else
                {
                        clrscr();
                        printf("Invalid  priority");
                }
       break;
    case 2:del();
       break;
    case 3:
       display();
       break;
    case 4:
       exit(0);
    default:
       printf("\nChoice is incorrect, Enter a correct choice");
    }
}
```

```
}

void insert(int x)
{
	switch(p)
	{
		case 1:if(r1==(n-1))
			printf("\n priority 1 overflow");
		else{
				if(fr1==-1)
					fr1=r1=0;
				else
					r1++;
				pq1[r1]=x;
		}
		break;
		case 2:if(r2==(n-1))
			printf("\n priority 2 overflow");
		else{
				if(fr2==-1)
					fr2=r2=0;
				else
					r2++;
				pq2[r2]=x;
		}
		break;
		case 3:if(r3==(n-1))
			printf("\n priority 3 overflow");
		else{
				if(fr3==-1)
					fr3=r3=0;
				else
					r3++;
```

```
                        pq3[r3]=x;
                }
                break;
}
}
void del()
{
        if(fr1==-1)
                printf("\n priority 1 underflow");
        else
        {
                printf("\n the deletedfrom pq1 : %d",pq1[fr1]);
                if(fr1==r1)
                        fr1=r1=-1;
                else
                        fr1++;
                getch();
                return;
        }

        if(fr2==-1)
                printf("\n priority 2 underflow");
        else
        {
                printf("\n the deletedfrom pq1 : %d",pq2[fr2]);
                if(fr2==r2)
                        fr2=r2=-1;
                else
                        fr2++;
                getch();
                return;
        }
```

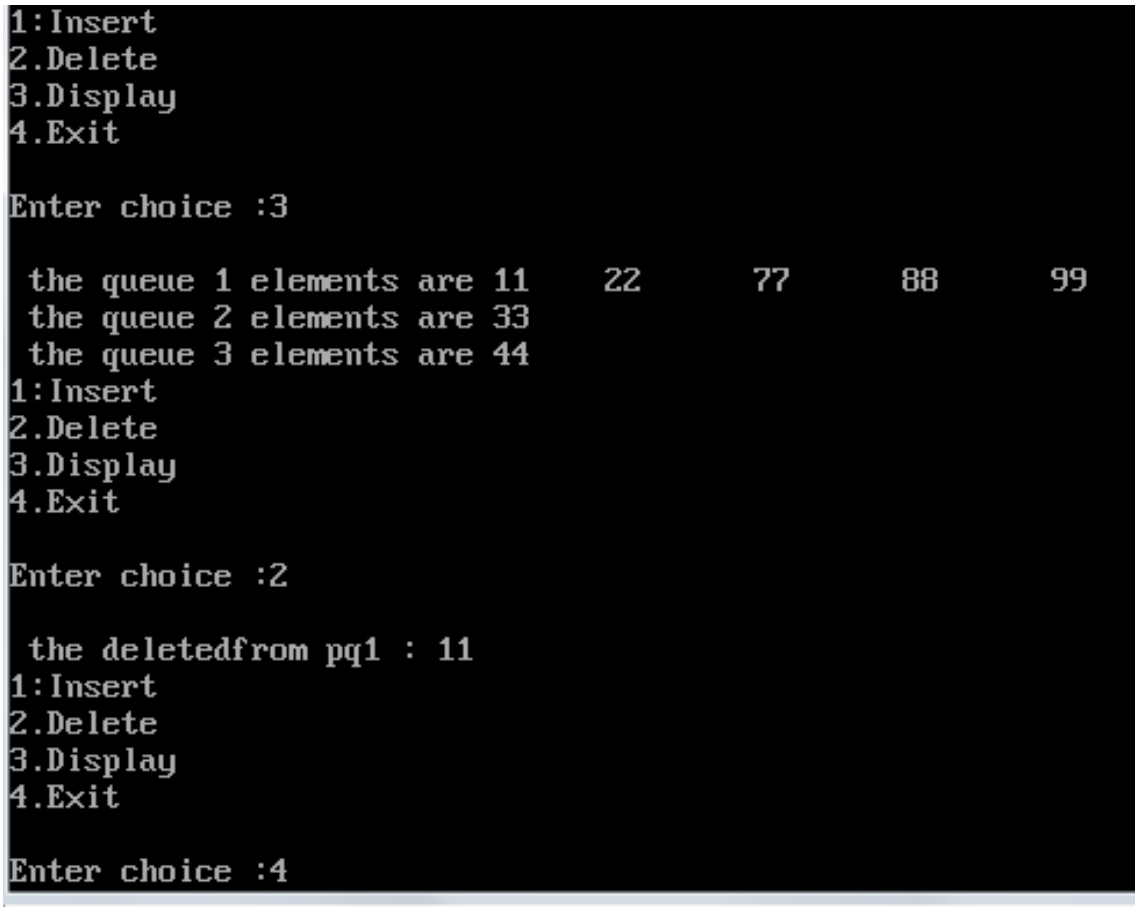
```
        if(fr3==-1)
                printf("\n priority 3 underflow");
        else
        {
                printf("\n the deletedfrom pq1 : %d",pq3[fr3]);
                if(fr3==r3)
                        fr3=r3=-1;
                else
                        fr3++;
                getch();
                return;
        }
}
 void display()
{
   if(fr1==-1)
        printf("\nQueue 1 is empty");
                else
                {
                        printf("\n the queue 1 elements are ");
                        for(i=fr1;i<=r1;i++)
                                printf("%d\t",pq1[i]);
                }

   if(fr2==-1)
        printf("\nQueue 2 is empty");
                else
                {
                        printf("\n the queue 2 elements are ");
                        for(i=fr2;i<=r2;i++)
                                printf("%d\t",pq2[i]);
                }
```

```
   if (fr3==-1)
        printf("\nQueue 3 is empty");
                else
                {
                                printf("\n the queue 3 elements are ");
                                for(i=fr3;i<=r3;i++)
                                        printf("%d\t",pq3[i]);
                }
}
```

Output :

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Program:     TC

 Enter size:5

1:Insert
2.Delete
3.Display
4.Exit

Enter choice :1
Enter queue element:11

 Enter priority(1,2,3):1

1:Insert
2.Delete
3.Display
4.Exit

Enter choice :_
```

```
 priority 1 overflow
1:Insert
2.Delete
3.Display
4.Exit

Enter choice :3

 the queue 1 elements are 11    22      77      88      99
 the queue 2 elements are 33
 the queue 3 elements are 44
1:Insert
2.Delete
3.Display
4.Exit

Enter choice :_
```


1:Insert
2.Delete
3.Display
4.Exit
Enter choice :3
the queue 1 elements are 11 22 77 88 99
the queue 2 elements are 33
the queue 3 elements are 44
1:Insert
2.Delete
3.Display
4.Exit
Enter choice :2
the deletedfrom pq1 : 11
1:Insert
2.Delete
3.Display
4.Exit
Enter choice :4