```
/* Program of circular queue using array*/

# include<stdio.h>

# define MAX 5


int cqueue_arr[MAX];

int front = -1;

int rear = -1;


main()

{

        int choice;

        while(1)

        {

                printf("1.Insert\n");

                printf("2.Delete\n");

                printf("3.Display\n");

                printf("4.Quit\n");

                printf("Enter your choice : ");

                scanf("%d",&choice);


                switch(choice)

                {

                case 1 :

                        insert();

                        break;

                case 2 :

                        del();

                        break;

                case 3:

                        display();

                        break;
                case 4:
                        exit(1);
                default:
                        printf("Wrong choice\n");
                }/*End of switch*/
        }/*End of while */
}/*End of main()*/

insert()
{
        int added_item;
        if((front == 0 && rear == MAX-1) || (front == rear+1))
        {
                printf("Queue Overflow \n");
                return;
        }
        if (front == -1)  /*If queue is empty */
        {
                front = 0;
                rear = 0;
        }
        else
                if(rear == MAX-1)/*rear is at last position of queue */
                        rear = 0;
                else
                        rear = rear+1;
        printf("Input the element for insertion in queue : ");
        scanf("%d", &added_item);
        cqueue_arr[rear] = added_item ;
}/*End of insert()*/
```

```
del()
{
	if (front == -1)
	{
		printf("Queue Underflow\n");
		return ;
	}
	printf("Element deleted from queue is : %d\n",cqueue_arr[front]);
	if(front == rear) /* queue has only one element */
	{
		front = -1;
		rear=-1;
	}
	else
		if(front == MAX-1)
			front = 0;
		else
			front = front+1;
}/*End of del() */

display()
{
	int front_pos = front,rear_pos = rear;
	if(front == -1)
	{
		printf("Queue is empty\n");
		return;
	}
	printf("Queue elements :\n");
	if( front_pos <= rear_pos )
		while(front_pos <= rear_pos)
		{
			printf("%d ",cqueue_arr[front_pos]);
			front_pos++;
		}
	else
	{
		while(front_pos <= MAX-1)
		{
			printf("%d ",cqueue_arr[front_pos]);
			front_pos++;
		}
		front_pos = 0;
		while(front_pos <= rear_pos)
		{
			printf("%d ",cqueue_arr[front_pos]);
			front_pos++;
		}
	}/*End of else */
	printf("\n");
}/*End of display() */
```

```
/* Program of queue using circular linked list*/
# include <stdio.h>
# include <malloc.h>

struct node
{
	int info;
	struct node *link;
```

```
}*rear=NULL;

main()
{
        int choice;
        while(1)
        {
                printf("1.Insert \n");
                printf("2.Delete \n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                 case 1:
                        insert();
                        break;
                 case 2:
                        del();
                        break;
                 case 3:
                        display();
                        break;
                 case 4:
                        exit();
                 default:
                        printf("Wrong choice\n");
                }/*End of switch*/
        }/*End of while*/
```

```
}/*End of main()*/

insert()
{
	int num;
	struct node *q,*tmp;
	printf("Enter the element for insertion : ");
	scanf("%d",&num);
	tmp= malloc(sizeof(struct node));
	tmp->info = num;

	if(rear == NULL) /*If queue is empty */
	{
		rear = tmp;
		tmp->link = rear;
	}
	else
	{
		tmp->link = rear->link;
		rear->link = tmp;
		rear = tmp;
	}
}/*End of insert()*/

del()
{
	struct node *tmp,*q;
	if(rear==NULL)
	{
		printf("Queue underflow\n");
		return;
```

```
        }
        if( rear->link == rear )  /*If only one element*/
        {
                tmp = rear;
                rear = NULL;
                free(tmp);
                return;
        }
        q=rear->link;
        tmp=q;
        rear->link = q->link;
        printf("Deleted element is %d\n",tmp->info);
        free(tmp);
}/*End of del()*/

display()
{
        struct node *q;
        if(rear == NULL)
        {
                printf("Queue is empty\n");
                return;
        }
        q = rear->link;
        printf("Queue is :\n");
        while(q != rear)
        {
                printf("%d ", q->info);
                q = q->link;
        }
        printf("%d\n",rear->info);
```

```
}/*End of display()*/


/* Program of input and output restricted dequeue using array*/
# include<stdio.h>
# define MAX 5


int deque_arr[MAX];
int left = -1;
int right = -1;


main()
{
        int choice;


        printf("1.Input restricted dequeue\n");
        printf("2.Output restricted dequeue\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);


        switch(choice)
        {
         case 1 :
                input_que();
                break;
         case 2:
                output_que();
                break;
         default:
                printf("Wrong choice\n");
        }/*End of switch*/
}/*End of main()*/
```

```
input_que()
{
        int choice;
        while(1)
        {
                printf("1.Insert at right\n");
                printf("2.Delete from left\n");
                printf("3.Delete from right\n");
                printf("4.Display\n");
                printf("5.Quit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                 case 1:
                        insert_right();
                        break;
                 case 2:
                        delete_left();
                        break;
                 case 3:
                        delete_right();
                        break;
                 case 4:
                        display_queue();
                        break;
                 case 5:
                        exit();
                 default:
                    printf("Wrong choice\n");
                }/*End of switch*/
        }/*End of while*/
}/*End of input_que() */

output_que()
{
        int choice;
        while(1)
        {
                printf("1.Insert at right\n");
                printf("2.Insert at left\n");
                printf("3.Delete from left\n");
                printf("4.Display\n");
                printf("5.Quit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                 case 1:
                        insert_right();
                        break;
                 case 2:
                        insert_left();
                        break;
                 case 3:
                        delete_left();
                        break;
                 case 4:
                        display_queue();
```

```
                                break;
                        case 5:
                                exit();
                        default:
                                printf("Wrong choice\n");
                        }/*End of switch*/
                }/*End of while*/
}/*End of output_que() */

insert_right()
{
        int added_item;
        if((left == 0 && right == MAX-1) || (left == right+1))
        {
                printf("Queue Overflow\n");
                return;
        }
        if (left == -1)  /* if queue is initially empty */
        {
                left = 0;
                right = 0;
        }
        else
        if(right == MAX-1)  /*right is at last position of queue */
                right = 0;
        else
                right = right+1;
        printf("Input the element for adding in queue : ");
        scanf("%d", &added_item);
        deque_arr[right] = added_item ;
}/*End of insert_right()*/
```

```
insert_left()
{
        int added_item;
        if((left == 0 && right == MAX-1) || (left == right+1))
        {
                printf("Queue Overflow \n");
                return;
        }
        if (left == -1)/*If queue is initially empty*/
        {
                left = 0;
                right = 0;
        }
        else
        if(left== 0)
                left=MAX-1;
        else
                left=left-1;
        printf("Input the element for adding in queue : ");
        scanf("%d", &added_item);
        deque_arr[left] = added_item ;
}/*End of insert_left()*/

delete_left()
{
        if (left == -1)
        {
                printf("Queue Underflow\n");
                return ;
        }
        printf("Element deleted from queue is : %d\n",deque_arr[left]);
        if(left == right) /*Queue has only one element */
        {
                left = -1;
                right=-1;
        }
        else
                if(left == MAX-1)
                        left = 0;
                else
                        left = left+1;
}/*End of delete_left()*/

delete_right()
{
        if (left == -1)
        {
                printf("Queue Underflow\n");
                return ;
        }
        printf("Element deleted from queue is : %d\n",deque_arr[right]);
        if(left == right) /*queue has only one element*/
        {
                left = -1;
                right=-1;
        }
        else
                if(right == 0)
                        right=MAX-1;
                else
                        right=right-1;
```

```
}/*End of delete_right() */

display_queue()
{
	int front_pos = left,rear_pos = right;
	if(left == -1)
	{
		printf("Queue is empty\n");
		return;
	}
	printf("Queue elements :\n");
	if( front_pos <= rear_pos )
	{
		while(front_pos <= rear_pos)
		{
			printf("%d ",deque_arr[front_pos]);
			front_pos++;
		}
	}
	else
	{
		while(front_pos <= MAX-1)
		{
			printf("%d ",deque_arr[front_pos]);
			front_pos++;
		}
		front_pos = 0;
		while(front_pos <= rear_pos)
		{
			printf("%d ",deque_arr[front_pos]);
			front_pos++;
```

```
                }
            }/*End of else */
            printf("\n");
}/*End of display_queue() */

/* Program of priority queue using linked list*/
# include<stdio.h>
# include<malloc.h>

struct node
{
            int priority;
            int info;
            struct node *link;
}*front = NULL;

main()
{
            int choice;
            while(1)
            {
                        printf("1.Insert\n");
                        printf("2.Delete\n");
                        printf("3.Display\n");
                        printf("4.Quit\n");
                        printf("Enter your choice : ");
                        scanf("%d", &choice);

                        switch(choice)
                        {
                         case 1:
```

```
                        insert();
                        break;
                case 2:
                        del();
                        break;
                case 3:
                        display();
                        break;
                case 4:
                        exit(1);
                default :
                        printf("Wrong choice\n");
                }/*End of switch*/
        }/*End of while*/
}/*End of main()*/

insert()
{
        struct node *tmp,*q;
        int added_item,item_priority;
        tmp = (struct node *)malloc(sizeof(struct node));
        printf("Input the item value to be added in the queue : ");
        scanf("%d",&added_item);
        printf("Enter its priority : ");
        scanf("%d",&item_priority);
        tmp->info = added_item;
        tmp->priority = item_priority;
        /*Queue is empty or item to be added has priority more than first item*/
        if( front == NULL || item_priority < front->priority )
        {
                tmp->link = front;
```

```
		front = tmp;
	}
	else
	{
		q = front;
		while( q->link != NULL && q->link->priority <= item_priority )
			q=q->link;
		tmp->link = q->link;
		q->link = tmp;
	}/*End of else*/
}/*End of insert()*/

del()
{
	struct node *tmp;
	if(front == NULL)
		printf("Queue Underflow\n");
	else
	{
		tmp = front;
		printf("Deleted item is %d\n",tmp->info);
		front = front->link;
		free(tmp);
	}
}/*End of del()*/

display()
{
	struct node *ptr;
	ptr = front;
	if(front == NULL)
```

```
                printf("Queue is empty\n");
        else
        {       printf("Queue is :\n");
                printf("Priority      Item\n");
                while(ptr != NULL)
                {
                        printf("%5d       %5d\n",ptr->priority,ptr->info);
                        ptr = ptr->link;
                }
        }/*End of else */
}/*End of display() */
```