Name – Shrinivas Hatyalikar
Div – CS-B
Roll no – 24

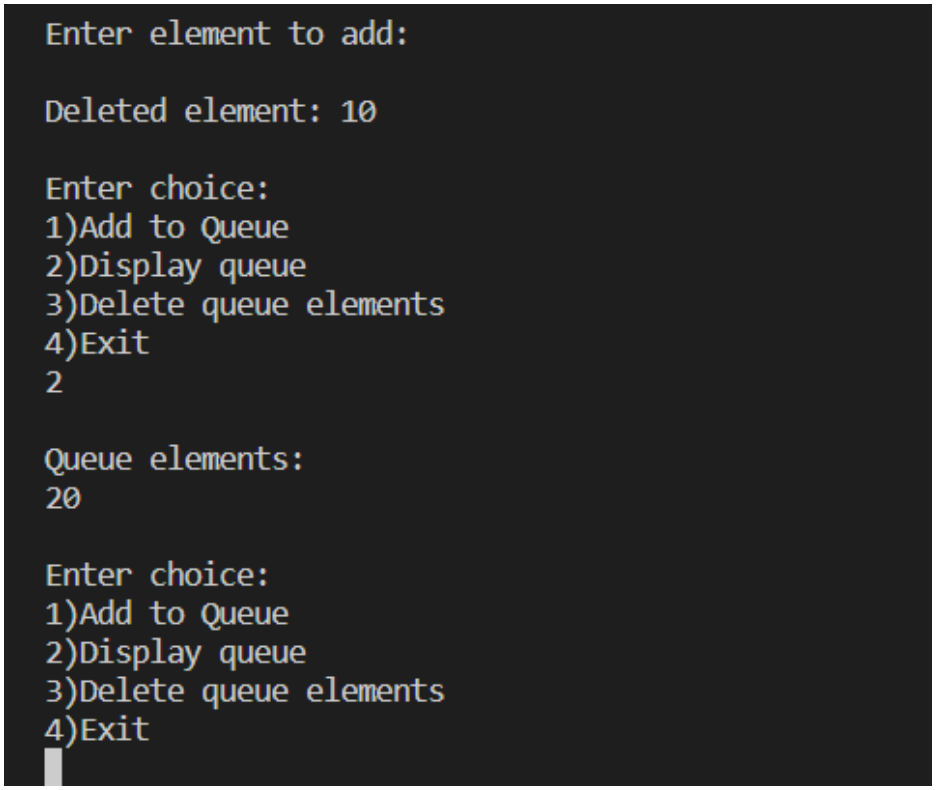
Linear Queue Using Array:

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

struct Queue{
    int size;
    int front;
    int rear;
    int *Q;
};

void create(struct Queue *q,int size){
    q->size=size;
    q->front=q->rear=-1;
    q->Q=(int *)malloc(q->size*sizeof(int));
}

void enqueue(struct Queue *q,int x ){
    if(q->rear==q->size-1){
        printf("Queue is FULL");
    }
    else{
        q->rear++;
        q->Q[q->rear]=x;
    }
}

int dequeue(struct Queue *q){
    int x=-1;
    if(q->rear==q->front){
        printf("Queue is EMPTY");
    }
    else{
```

```
        q->front++;
        x=q->Q[q->front];
    }
    return x;
}

void display(struct Queue q){
    int i=0;
    for(i=q.front+1;i<=q.rear;i++){
        printf("%d ",q.Q[i]);
    }
    printf("\n");
}

int main(){
    struct Queue q;
    create(&q,5);
    int x,choice;
    while(1){
        printf("\nEnter choice:\n1)Add to Queue\n2)Display queue\n3)Delete queue
elements\n4)Exit\n");
        scanf("%d",&choice);
        switch(choice){
            case 1: {
                printf("\nEnter element to add:\n");
                scanf("%d",&x);
                enqueue(&q,x);
                break;
            }
            case 2: {
                printf("\nQueue elements:\n");
                display(q);
                break;
            }
            case 3: {
                printf("\nDeleted element: %d\n",dequeue(&q));
                break;
            }
            case 4: {
                exit(0);
```

```
        }
        default: {
            printf("Invalid choice.\n");
        }
    }
  }
}
```

```
Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1

Enter element to add:
10

Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1
```



Circular Queue Using Array:

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

struct queue{
    int size;
    int front,rear;
    int *Q;
};

void create(struct queue *q,int size){
    q->size=size;
    q->front=q->rear=0;
    q->Q=(int *)malloc(q->size*sizeof(int));
}
```

```
void enqueue(struct queue *q,int x){
    if((q->rear+1)%q->size==q->front){
        printf("\nQUEUE is FULL");
    }
    else{
        q->rear=(q->rear+1)%q->size;
        q->Q[q->rear]=x;
    }
}

int dequeue(struct queue *q){
    int x=-1;
    if(q->rear==q->front){
        printf("\nQUEUE is EMPTY!!");
    }
    else{
        q->front=(q->front+1)%q->size;
        x=q->Q[q->front];
    }
    return x;
}

void display(struct queue q){
    int i=q.front+1;
    do{
        printf("%d\n",q.Q[i]);
        i=(i+1)%q.size;
    }while(i!=(q.rear+1)%q.size);
    printf("\n");
}

int main(){
    struct queue q;
    create(&q,5);
    int x,choice;
    while(1){
        printf("\nEnter choice:\n1)Add to Queue\n2)Display queue\n3)Delete queue
elements\n4)Exit\n");
        scanf("%d",&choice);
        switch(choice){
```

```
            case 1: {
                printf("\nEnter element to add:\n");
                scanf("%d",&x);
                enqueue(&q,x);
                break;
            }
            case 2: {
                printf("\nQueue elements:\n");
                display(q);
                break;
            }
            case 3: {
                printf("\nDeleted element: %d\n",dequeue(&q));
                break;
            }
            case 4: {
                exit(0);
            }
            default: {
                printf("\nInvalid choice.\n");
            }
        }
    }
}
```

```
Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1

Enter element to add:
10

Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1

Enter element to add:
20
```

```
Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
2

Queue elements:
10
20
```

```
Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
3

Deleted element: 10

Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
2

Queue elements:
20
```

Linear Queue Using LinkedList:

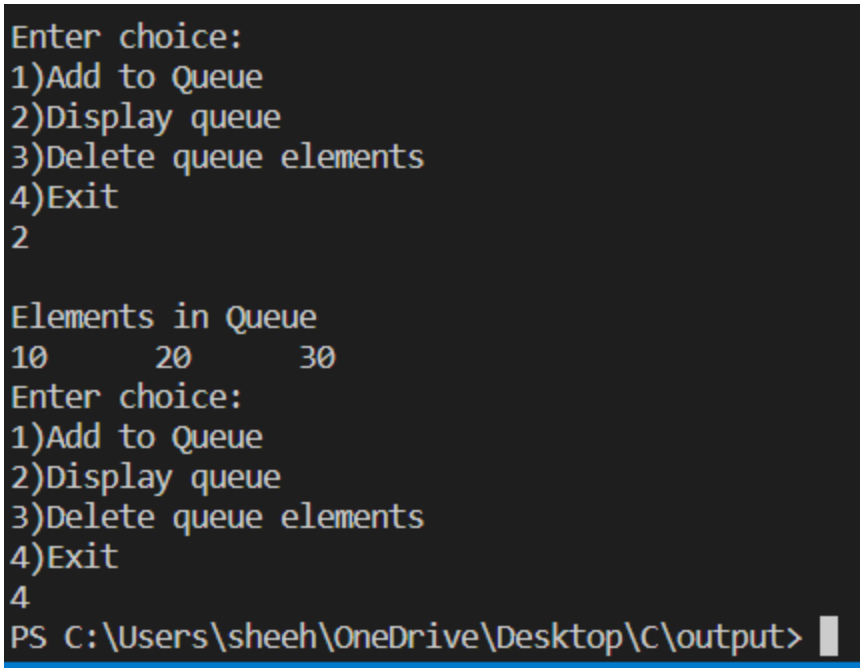
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *front;
struct node *rear;

void insert()
{
    struct node *ptr;
    int item;

    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
```

```
    {
        printf("\nOVERFLOW\n");
        return;
    }
    else
    {
        printf("\nEnter value?\n");
        scanf("%d",&item);
        ptr -> data = item;
        if(front == NULL)
        {
            front = ptr;
            rear = ptr;
            front -> next = NULL;
            rear -> next = NULL;
        }
        else
        {
            rear -> next = ptr;
            rear = ptr;
            rear->next = NULL;
        }
    }
}
void delete ()
{
    struct node *ptr;
    if(front == NULL)
    {
        printf("\nUNDERFLOW\n");
        return;
    }
    else
    {
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}
void display()
```

```
{
   struct node *ptr;
   ptr = front;
   if(front == NULL)
   {
      printf("\nEmpty queue\n");
   }
   else
   {   printf("\nElements in Queue\n");
      while(ptr != NULL)
      {
         printf("%d\t",ptr -> data);
         ptr = ptr -> next;
      }
   }
}

void main ()
{
   int choice;
   while(1){
      printf("\nEnter choice:\n1)Add to Queue\n2)Display queue\n3)Delete queue
elements\n4)Exit\n");
      scanf("%d",&choice);
      switch(choice){
         case 1: {
            insert();
            break;
         }
         case 2: {
            display();
            break;
         }
         case 3: {
            delete();
            break;
         }
         case 4: {
            exit(0);
         }
```

```
        default: {
            printf("\nInvalid choice.\n");
        }
    }
  }
}
```

```
Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1

Enter value?
10

Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1

Enter value?
20

Enter choice:
1)Add to Queue
2)Display queue
3)Delete queue elements
4)Exit
1

Enter value?
30
```



Circular Linear Queue Using LinkedList:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node * next;
}* front=NULL,*rear=NULL;
void display(){
if(front==NULL){
printf("Queue is Empty");
}
else{
struct node *temp=front;
while(temp!=rear){
printf("%d ",temp->data);
temp=temp->next;
}
```

```
printf("%d ",temp->data);
}
}
void enqueue(int x){
if(rear==NULL){
rear=(struct node *)malloc(sizeof(struct node));
rear->data=x;
rear->next=NULL;
front=rear;
}
else{
struct node * temp=(struct node *)malloc(sizeof(struct node));
temp->data=x;
temp->next=front;
rear->next=temp;
rear=temp;;
}
}
int dequeue(){
int x;
if(front==NULL){
printf("Queue is empty");
}
else{
struct node * temp=front;
x=temp->data;
front=front->next;
free(temp);
}
return x;
}
int menu()
{
printf("\nChoose from below option -\n");
int opt;
printf("1 : Enqueue element \n2 : Dequeue element \n3 : Print the Queue\n");
printf("\nEnter the option- ");
scanf(" %d", &opt);
return opt;
}
```

```
int main()
{
char more = 'y';
while (more == 'y')
{
int opt = menu();
if (opt == 1)
{
printf("How many elements ? ");
int n;
scanf(" %d", &n);
for (int i = 0; i < n; i++)
{
int ele1;
printf("Data : ");
scanf(" %d", &ele1);
enqueue(ele1);
}
}
else if (opt == 2)
{
int x=dequeue();
printf("Deleted Element is %d\n", x);
// delete only 1 element.
}
else if (opt == 3)
{
display();
}
printf("\nMore tasks to do ? (y/n) ");
scanf(" %c", &more);
}
}
```

```
Choose from below option -
1 : Enqueue element
2 : Dequeue element
3 : Print the Queue

Enter the option- 1
How many elements ? 4
Data : 10
Data : 20
Data : 30
Data : 40

More tasks to do ? (y/n) y

Choose from below option -
1 : Enqueue element
2 : Dequeue element
3 : Print the Queue

Enter the option- 3
10 20 30 40
More tasks to do ? (y/n) n
PS C:\Users\sheeh\OneDrive\Desktop\C\output>
```