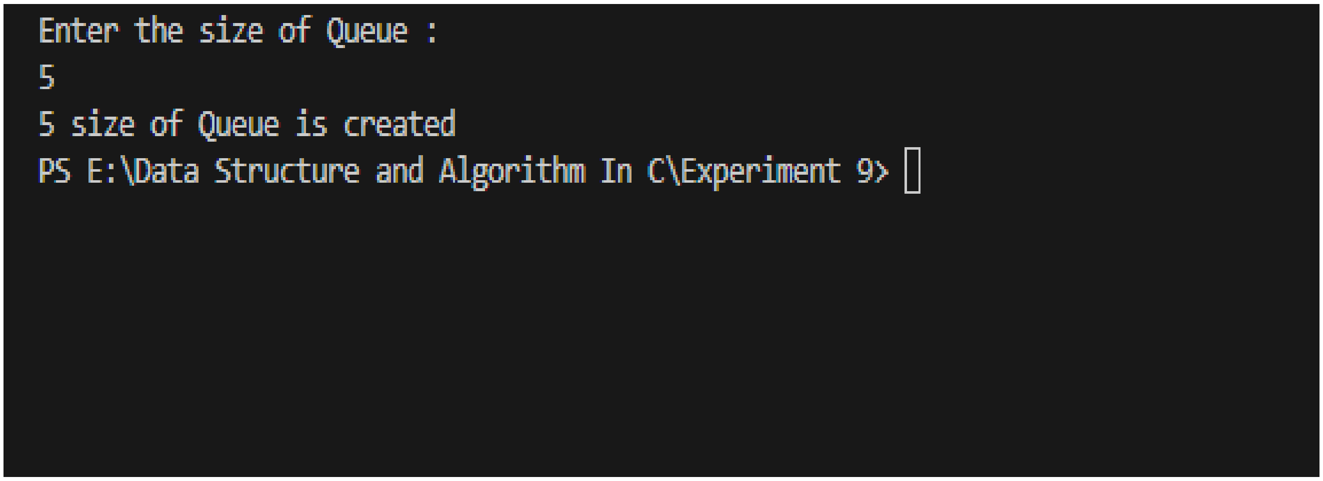
EXPERIMENT NO. 9

Ques 1 :- Write a program to create a Queue?

```
#include<stdio.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));
   printf("%d size of Queue is created",q->size);
}
int main()
{
   int size;
   struct Queue q;
```

```
    printf("Enter the size of Queue :\n");
    scanf("%d",&size);
    create(&q,size);
    return 0;
}
```

Output of the Code :-

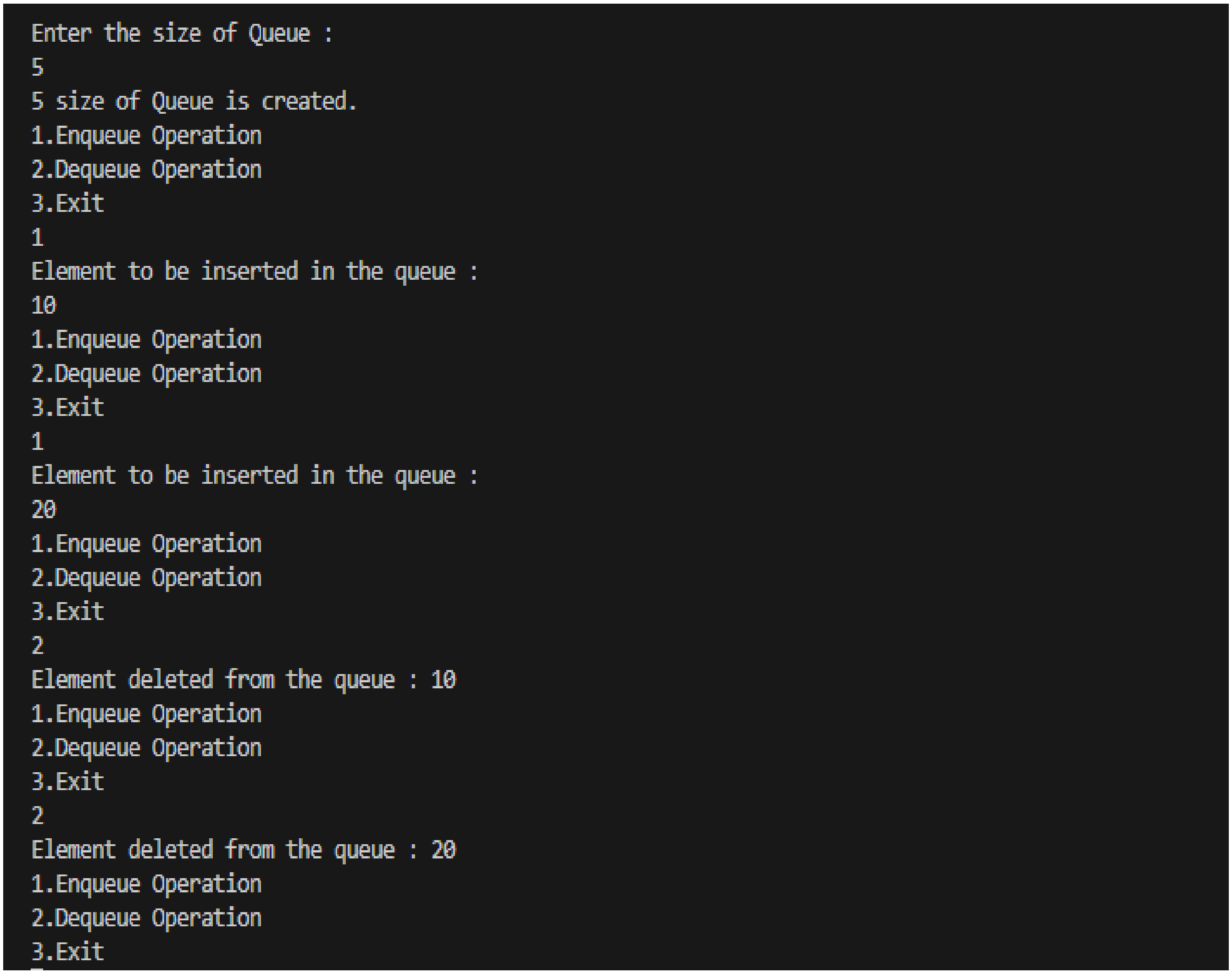


```
Enter the size of Queue :
5
5 size of Queue is created
PS E:\Data Structure and Algorithm In C\Experiment 9>
```

Ques 2 :- Write a program to perform Enqueue and Deque operations on Queue?

```
#include<stdio.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));
    printf("%d size of Queue is created.\n",q->size);
}
void enqueue(struct Queue *q)
{
    int insert_item;
    if(q->rear==q->size-1)
    {
        printf("Queue is Overflow");
    }
    else
    {
        q->front=0;
        printf("Element to be inserted in the queue :\n");
        scanf("%d",&insert_item);
        q->rear++;
        q->Q[q->rear]=insert_item;
    }
}
void dequeue(struct Queue *q)
{
    if(q->rear==-1||q->front>q->rear)
    {
        printf("Queue is underflow\n");
    }
    else
    {
        printf("Element deleted from the queue : %d\n",q->Q[q->front]);
        q->front++;
    }
}
int main()
{
    int size;
    struct Queue q;
    printf("Enter the size of Queue :\n");
    scanf("%d",&size);
    create(&q,size);
    int ch;
    while(1)
    {
        printf("1.Enqueue Operation\n2.Dequeue Operation\n3.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: enqueue(&q);
            break;
            case 2: dequeue(&q);
            break;
            case 3 :exit(0);
            default:
            printf("Incorrect choice\n");
        }
    }
    return 0;
}
```

Output of the Code :-



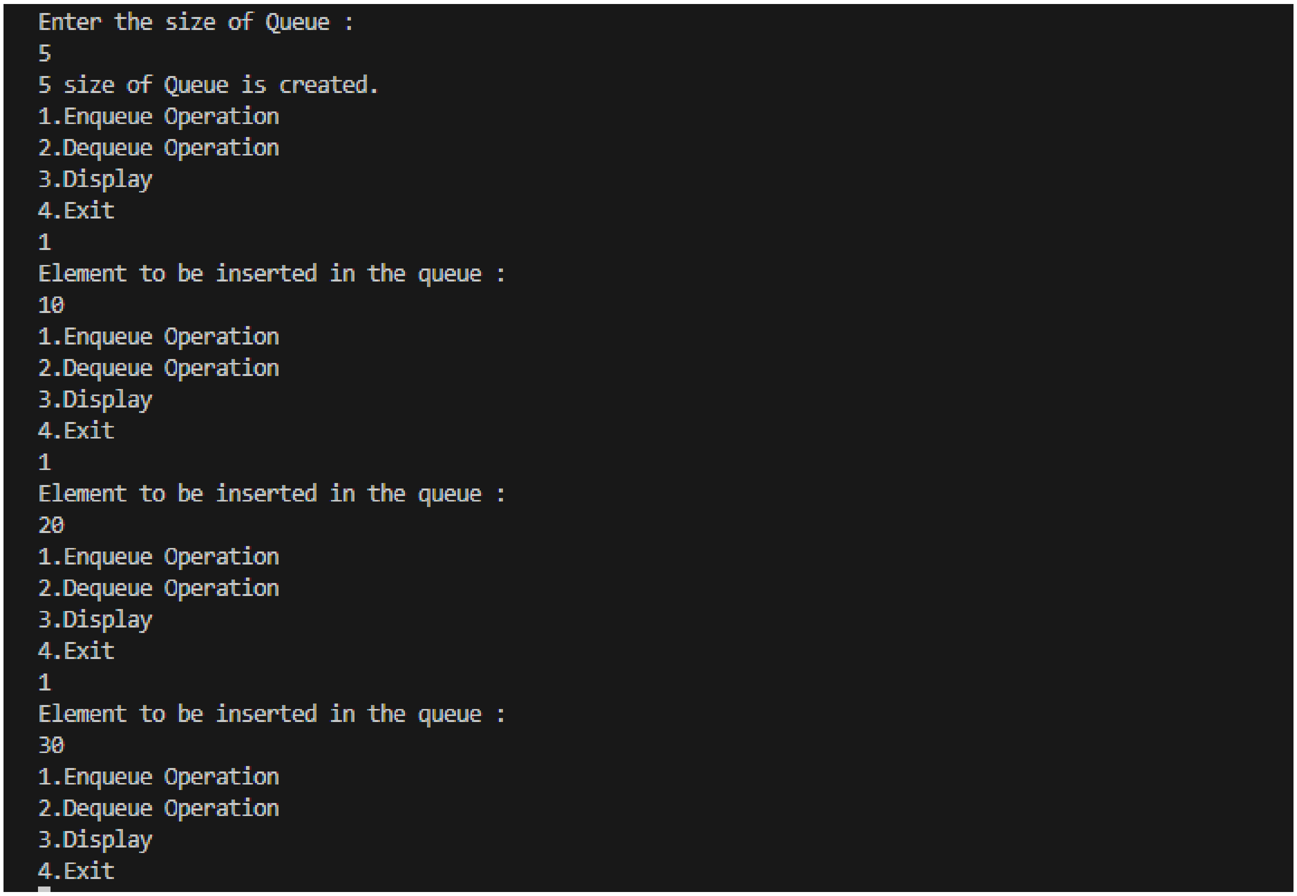
```
Enter the size of Queue :
5
5 size of Queue is created.
1.Enqueue Operation
2.Dequeue Operation
3.Exit
1
Element to be inserted in the queue :
10
1.Enqueue Operation
2.Dequeue Operation
3.Exit
1
Element to be inserted in the queue :
20
1.Enqueue Operation
2.Dequeue Operation
3.Exit
2
Element deleted from the queue : 10
1.Enqueue Operation
2.Dequeue Operation
3.Exit
2
Element deleted from the queue : 20
1.Enqueue Operation
2.Dequeue Operation
3.Exit
```

Ques 3 :- Write a program to traverse the queue and print its elements?

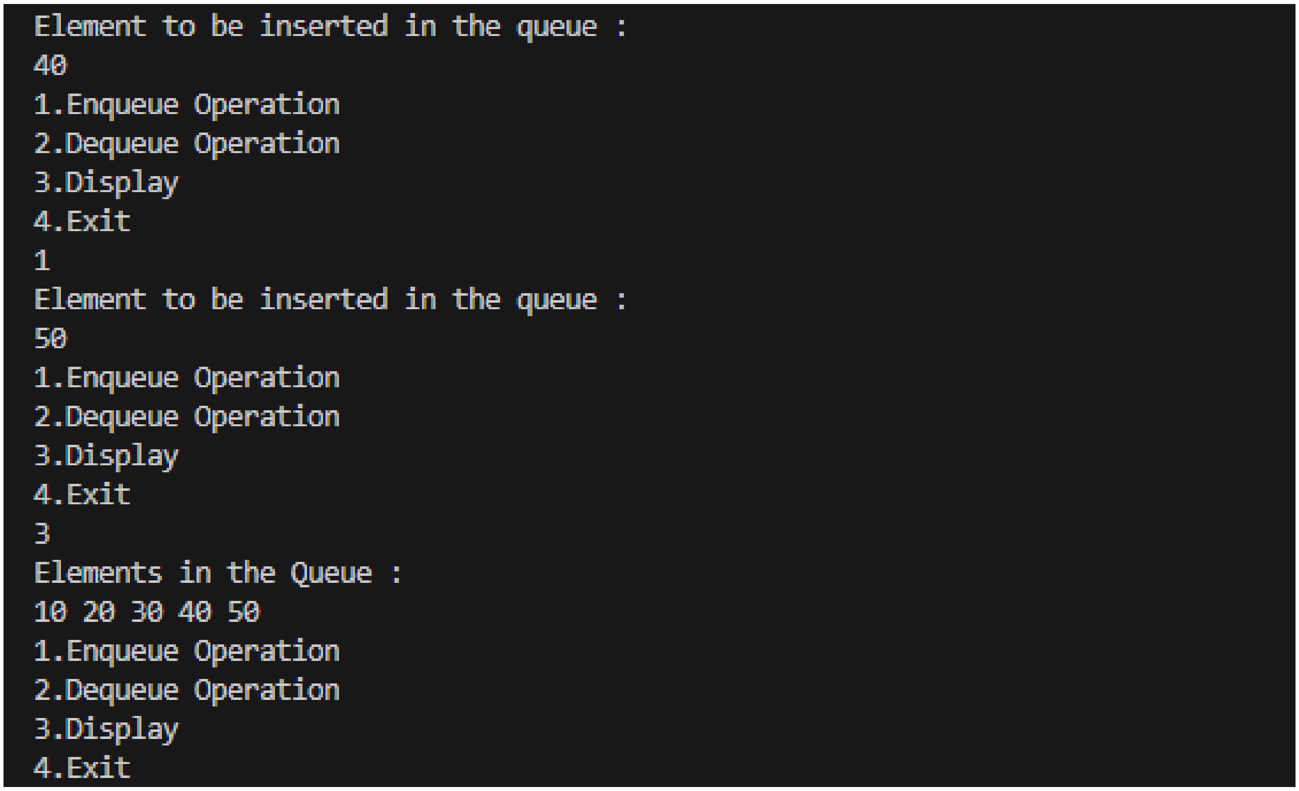
```
#include<stdio.h>

#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));
    printf("%d size of Queue is created.\n",q->size);
}
void enqueue(struct Queue *q)
{
    int insert_item;
    if(q->rear==q->size-1)
    {
        printf("Queue is Overflow");
    }
    else
    {
        q->front=0;
        printf("Element to be inserted in the queue :\n");
        scanf("%d",&insert_item);
        q->rear++;
        q->Q[q->rear]=insert_item;
    }
}
void dequeue(struct Queue *q)
{
    if(q->rear==-1||q->front>q->rear)
    {
        printf("Queue is underflow\n");
    }
    else
    {
        printf("Element deleted from the queue : %d\n",q->Q[q->front]);
        q->front++;
    }
}
void display(struct Queue *q)
{
     if(q->rear==-1)
    {
        printf("Empty Queue \n");
    }
    else
    {
        printf("Elements in the Queue :\n");
        for(int i=q->front;i<=q->rear;i++)
        {
            printf("%d ",q->Q[i]);
        }
        printf("\n");
    }
}
int main()
{
    int size;
    struct Queue q;
    printf("Enter the size of Queue :\n");
    scanf("%d",&size);
    create(&q,size);
    int ch;
    while(1)
    {
        printf("1.Enqueue Operation\n2.Dequeue
Operation\n3.Display\n4.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: enqueue(&q);
            break;
            case 2: dequeue(&q);
            break;
            case 3: display(&q);
            break;
            case 4 :exit(0);
            default:
            printf("Incorrect choice\n");
        }
    }
    return 0;
}
```

Output of the Code :-



```
Enter the size of Queue :
5
5 size of Queue is created.
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
10
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
20
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
30
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
```



```
Element to be inserted in the queue :
40
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
50
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
3
Elements in the Queue :
10 20 30 40 50
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
```

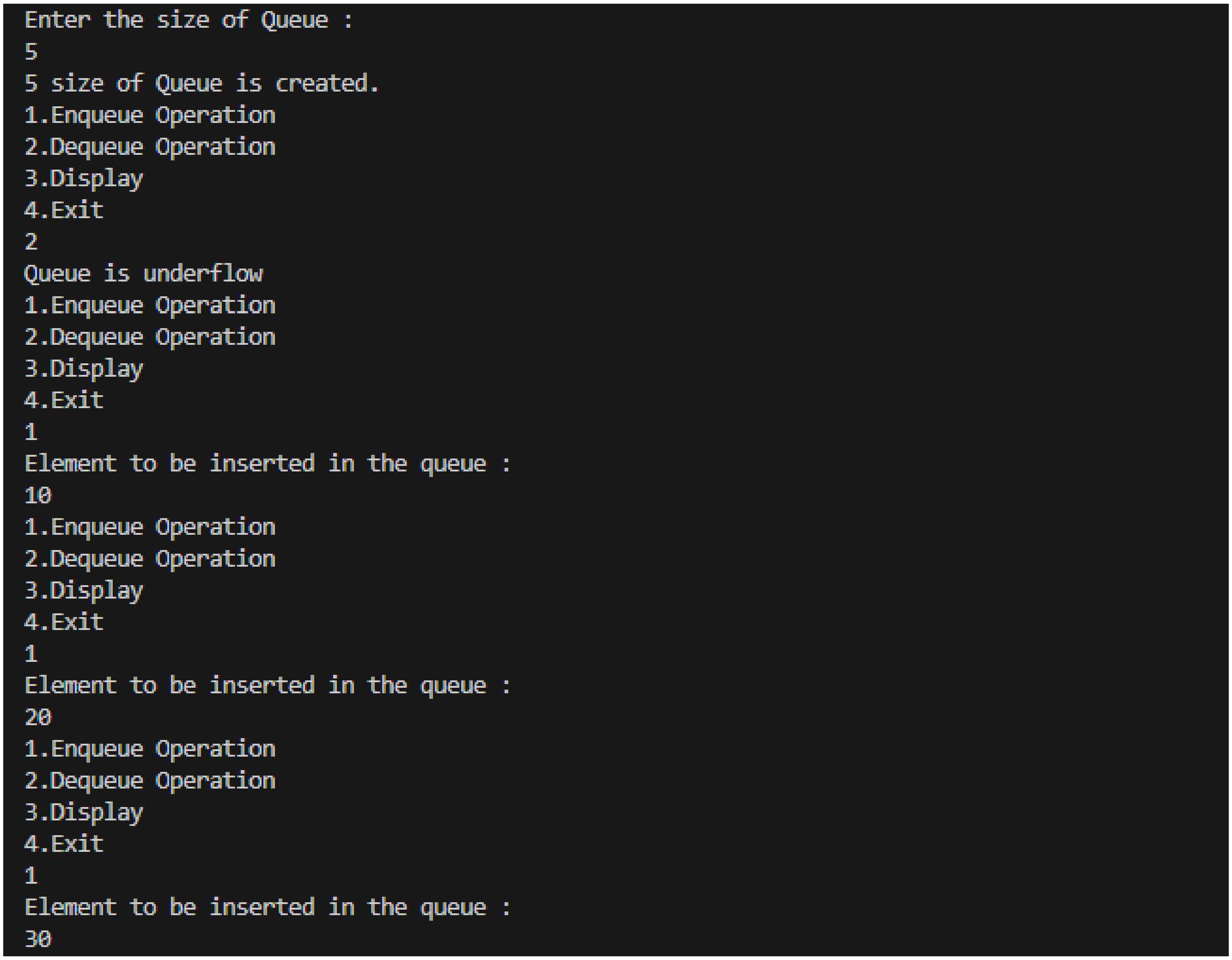
Ques 4 :- Write the program to print underflow and overflow when desired conditions are not met?

```
#include<stdio.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));

   printf("%d size of Queue is created.\n",q->size);

}

void enqueue(struct Queue *q)

{

   int insert_item;

   if(q->rear==q->size-1)
```

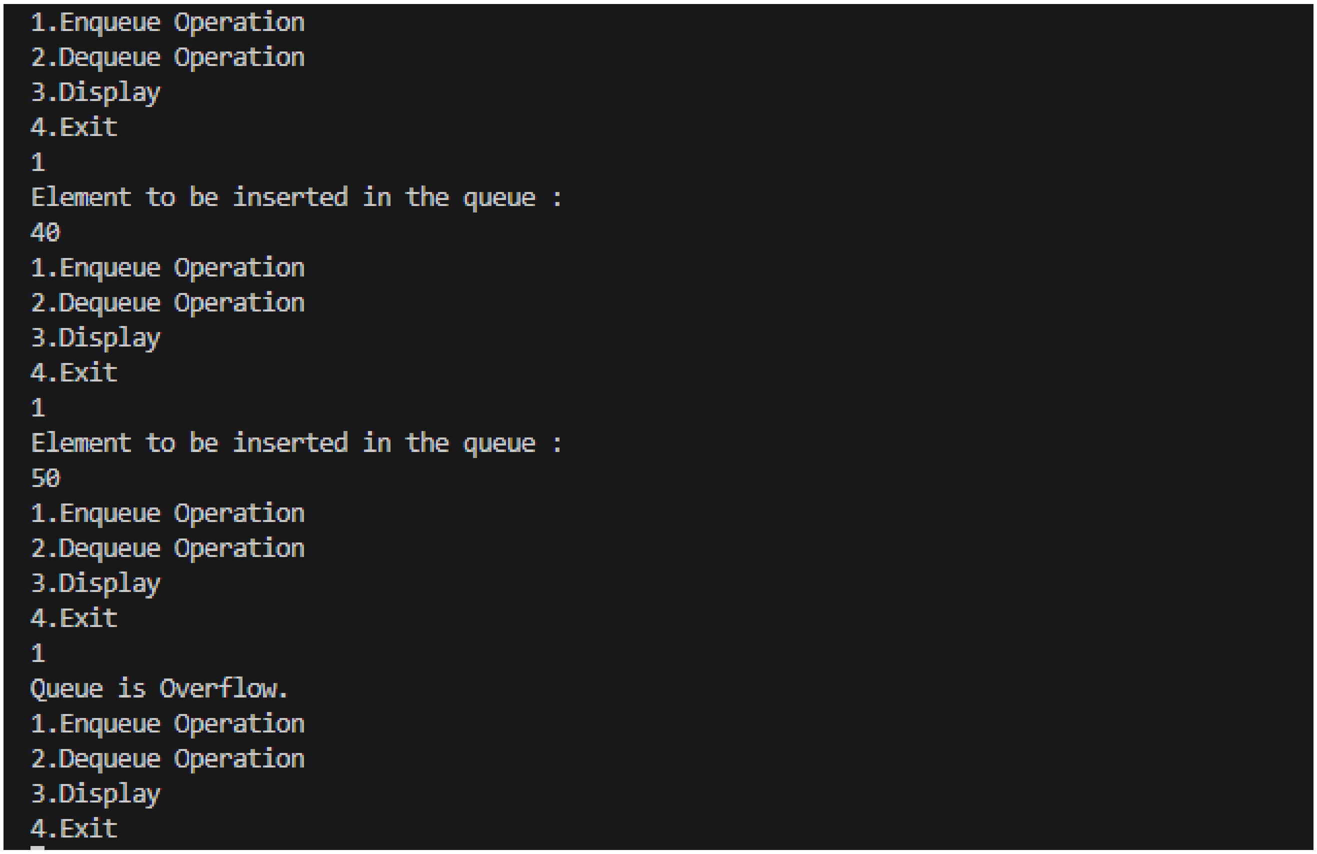
```
  {
    printf("Queue is Overflow");
  }
  else
  {
    q->front=0;
    printf("Element to be inserted in the queue :\n");
    scanf("%d",&insert_item);
    q->rear++;
    q->Q[q->rear]=insert_item;
  }
}
void dequeue(struct Queue *q)
{
  if(q->rear==-1||q->front>q->rear)
  {
    printf("Queue is underflow\n");
  }
  else
  {
    printf("Element deleted from the queue : %d\n",q->Q[q->front]);
    q->front++;
    }
}
void display(struct Queue *q)
{
     if(q->rear==-1)
    {
        printf("Empty Queue \n");
    }
    else
    {
        printf("Elements in the Queue :\n");
        for(int i=q->front;i<=q->rear;i++)
        {
            printf("%d ",q->Q[i]);
        }
        printf("\n");
    }
}
int main()
{
    int size;
    struct Queue q;
    printf("Enter the size of Queue :\n");
    scanf("%d",&size);
    create(&q,size);
    int ch;
    while(1)
    {
        printf("1.Enqueue Operation\n2.Dequeue Operation\n3.Display\n4.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: enqueue(&q);
            break;
            case 2: dequeue(&q);
            break;
            case 3: display(&q);
            break;
            case 4 :exit(0);
            default:
            printf("Incorrect choice\n");
        }
    }
    return 0;
```

```
}
```

Output of the Code :-



```
Enter the size of Queue :
5
5 size of Queue is created.
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
2
Queue is underflow
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
10
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
20
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
30
```



```
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
40
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Element to be inserted in the queue :
50
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
1
Queue is Overflow.
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Exit
```

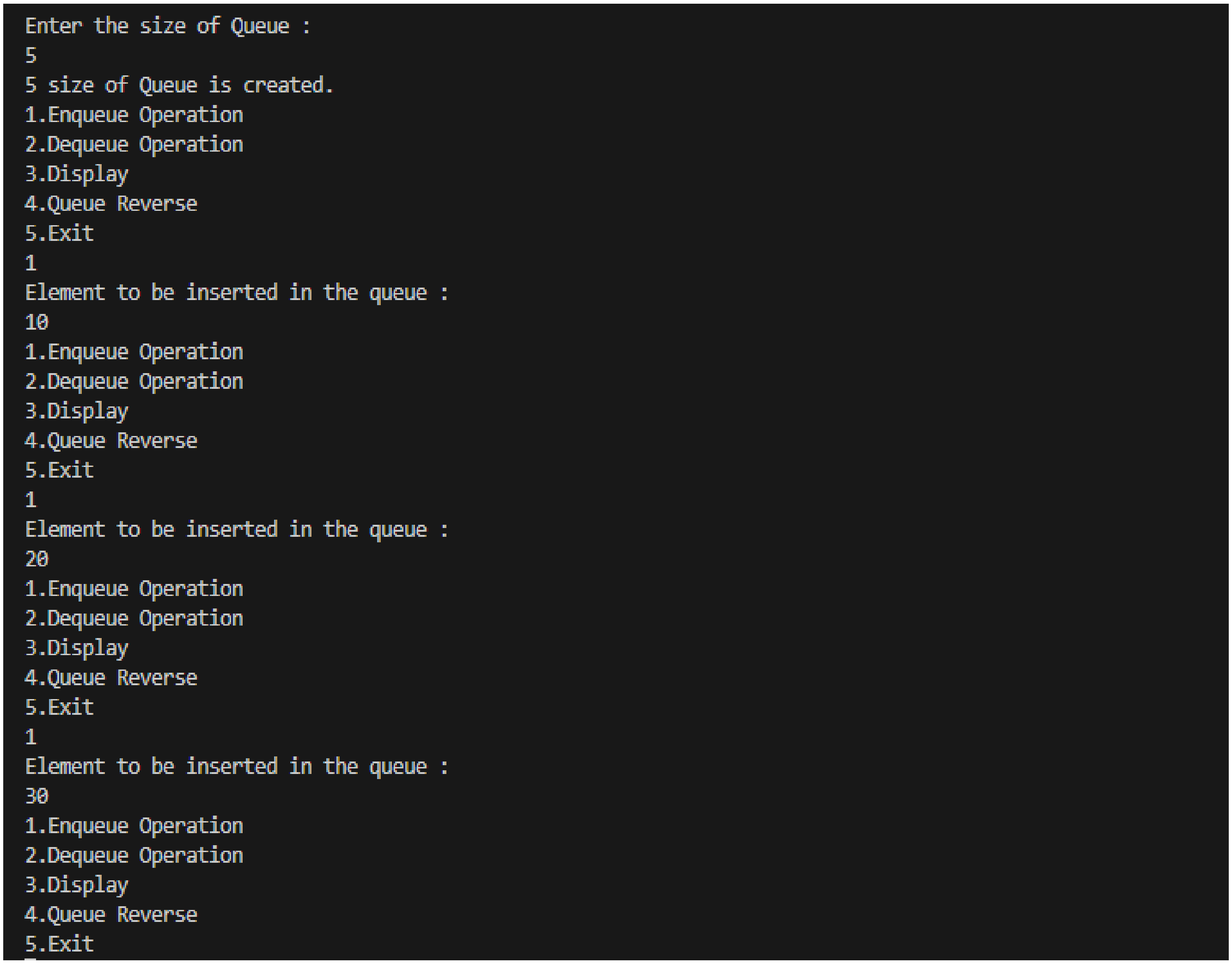
Ques 5 :- Write a program to reverse the elements of Queue using Recursion?

```
#include<stdio.h>

#include<stdlib.h>

struct Queue

{

   int size;

   int front;

   int rear;

   int *Q;
```

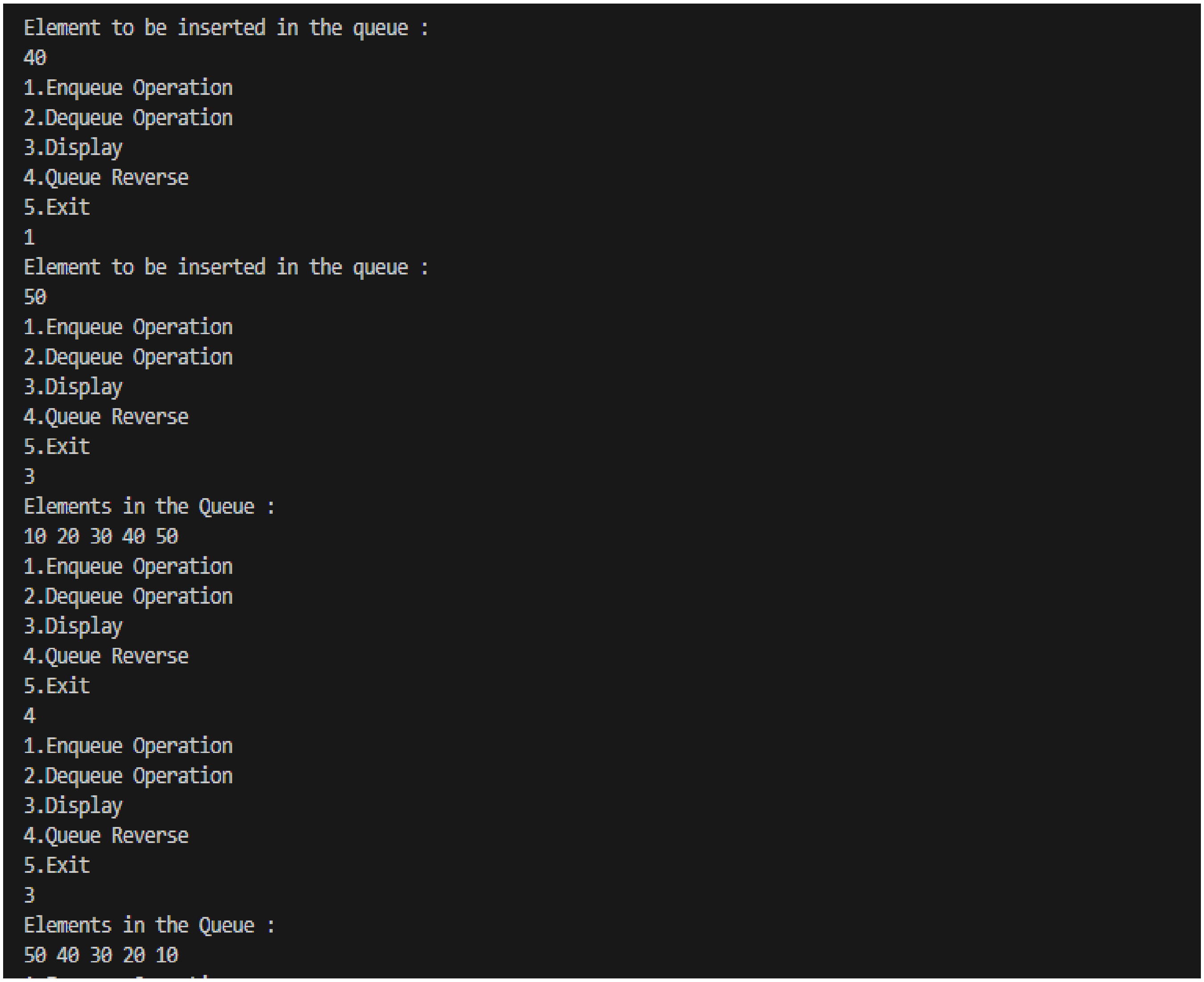
```
};
void create(struct Queue *q,int size)
{
   q->size=size;
   q->front=q->rear=-1;
   q->Q=(int*)malloc(q->size*sizeof(int));
   printf("%d size of Queue is created.\n",q->size);
}
void enqueue(struct Queue *q)
{
   int insert_item;
   if(q->rear==q->size-1)
   {
      printf("Queue is Overflow.\n");
   }
   else
   {
      q->front=0;
      printf("Element to be inserted in the queue :\n");
      scanf("%d",&insert_item);
      q->rear++;
      q->Q[q->rear]=insert_item;
```

```
    }
}
void dequeue(struct Queue *q)
{
    if(q->rear==-1||q->front>q->rear)
    {
        printf("Queue is underflow\n");
    }
    else
    {
        printf("Element deleted from the queue : %d\n",q->Q[q->front]);
        q->front++;
    }
}
void display(struct Queue *q)
{
     if(q->rear==-1)
    {
        printf("Empty Queue \n");
    }
    else
    {
        printf("Elements in the Queue :\n");
        for(int i=q->front;i<=q->rear;i++)
        {
            printf("%d ",q->Q[i]);
        }
        printf("\n");
    }
}
void queuereverse(struct Queue q, int *x)
{
    if(q.front>q.rear)
    return;
    int temp=q.Q[q.front];
    q.front++;
    queuereverse(q,x);
    q.Q[(*x)++]=temp;
}
int main()
{
    int size;
    struct Queue q;
    printf("Enter the size of Queue :\n");
    scanf("%d",&size);
    create(&q,size);
    int ch;
    while(1)
    {
        printf("1.Enqueue Operation\n2.Dequeue
Operation\n3.Display\n4.Queue Reverse\n5.Exit\n");
        scanf("%d",&ch);
        int x=0;
        switch(ch)
        {
            case 1: enqueue(&q);
            break;
            case 2: dequeue(&q);
            break;
            case 3: display(&q);
            break;
            case 4: queuereverse(q,&x);
            break;
            case 5 :exit(0);
            default:
            printf("Incorrect choice\n");
        }
    }
    return 0;
}
```

Output of the Code :-



```
Enter the size of Queue :
5
5 size of Queue is created.
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
1
Element to be inserted in the queue :
10
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
1
Element to be inserted in the queue :
20
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
1
Element to be inserted in the queue :
30
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
```



```
Element to be inserted in the queue :
40
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
1
Element to be inserted in the queue :
50
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
3
Elements in the Queue :
10 20 30 40 50
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
4
1.Enqueue Operation
2.Dequeue Operation
3.Display
4.Queue Reverse
5.Exit
3
Elements in the Queue :
50 40 30 20 10
```