8.1.2024

3. Write a program to stimulate the working of the queue of integers using an array. Provide the following operations:- Insert, Delete, Display. The program should print appropriate message for overflow and underflow condition.

```
# include<stdio.h>
# include <conio.h>
# include<process.h>
# define QUE_SIZE 5
int item, front =0, rear=-1, q[15];

void insert()
 {
  if (rear== QUE_SIZE-1)
 {
  printf("Queue overflow\n");
  return;
 }
  rear= rear+1;
  q[rear]=item;
 }
 int delete front()
 {
  if(front >rear) return-1;
  return q[front++];
 }
void displayQ()
 {
  int i;
 if (front> rear)
{
printf(" queue is empty\n");
return;
}
```

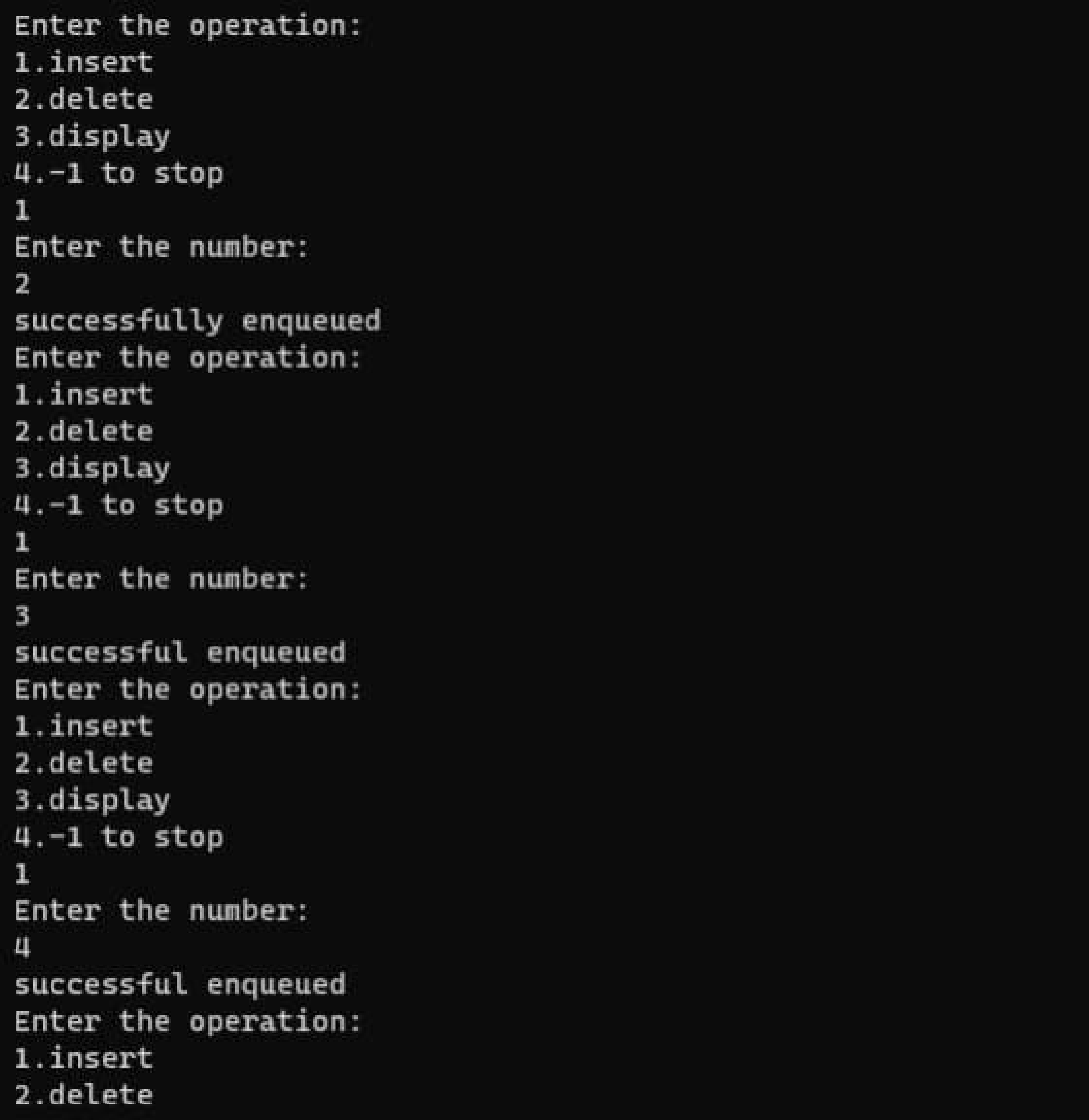
```
print f ("content of queue\n");
for(i=front; i<= rear; i++){
{
print f ("%d\n", q[i]);
 }
}
void main()
{
 int choice;
 clrscr();
 for(;;)
 {
  print f ("\n 1: insert rear\n 2: delete front\n 3: display\n
             4: exit\n");
  print f ("Enter the choice\n");
  scan f ("%d", &choice);
  switch (choice)
  {
  case 1: print f (" Enter the item to be inserted\n");
          scan f ("%d", &item);
          insert rear();
          break;
  case 2: item = delete front();
          if(item==-1)
          print f ("Queue is empty\n");
          else
          print f ("Item deleted = %d\n", item);
          break;
  case 3: display Q();
          break;
    default: exit(0);
   }
   }
   getch();
   }
```

```
         else
         printf("item deleted = %d\n", item);
         break;
     case 3: displayQ();
         break;
     default: exit(0);
     }
   }
   getch();
}
```

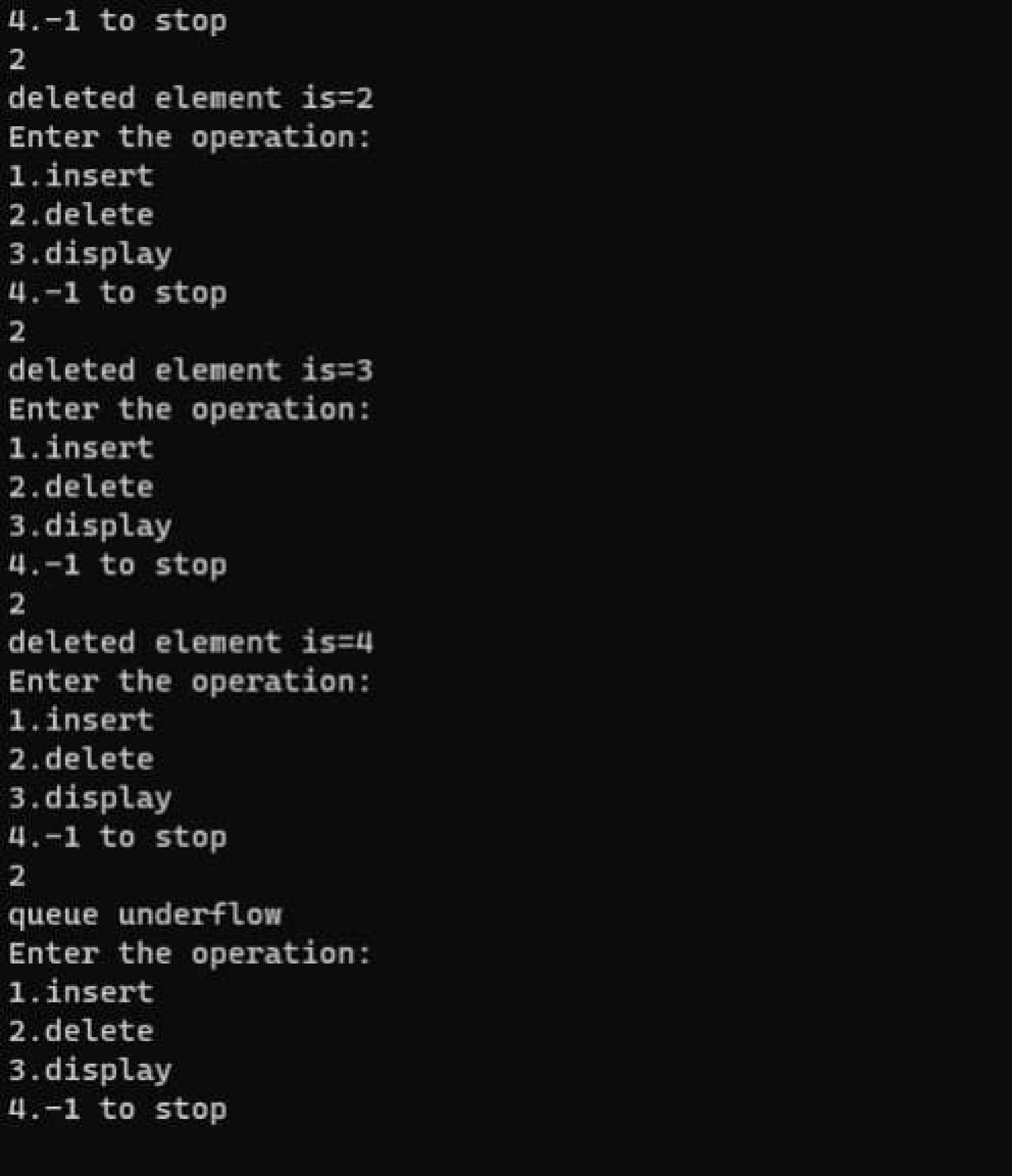
Both o/p's

OUTPUT :-

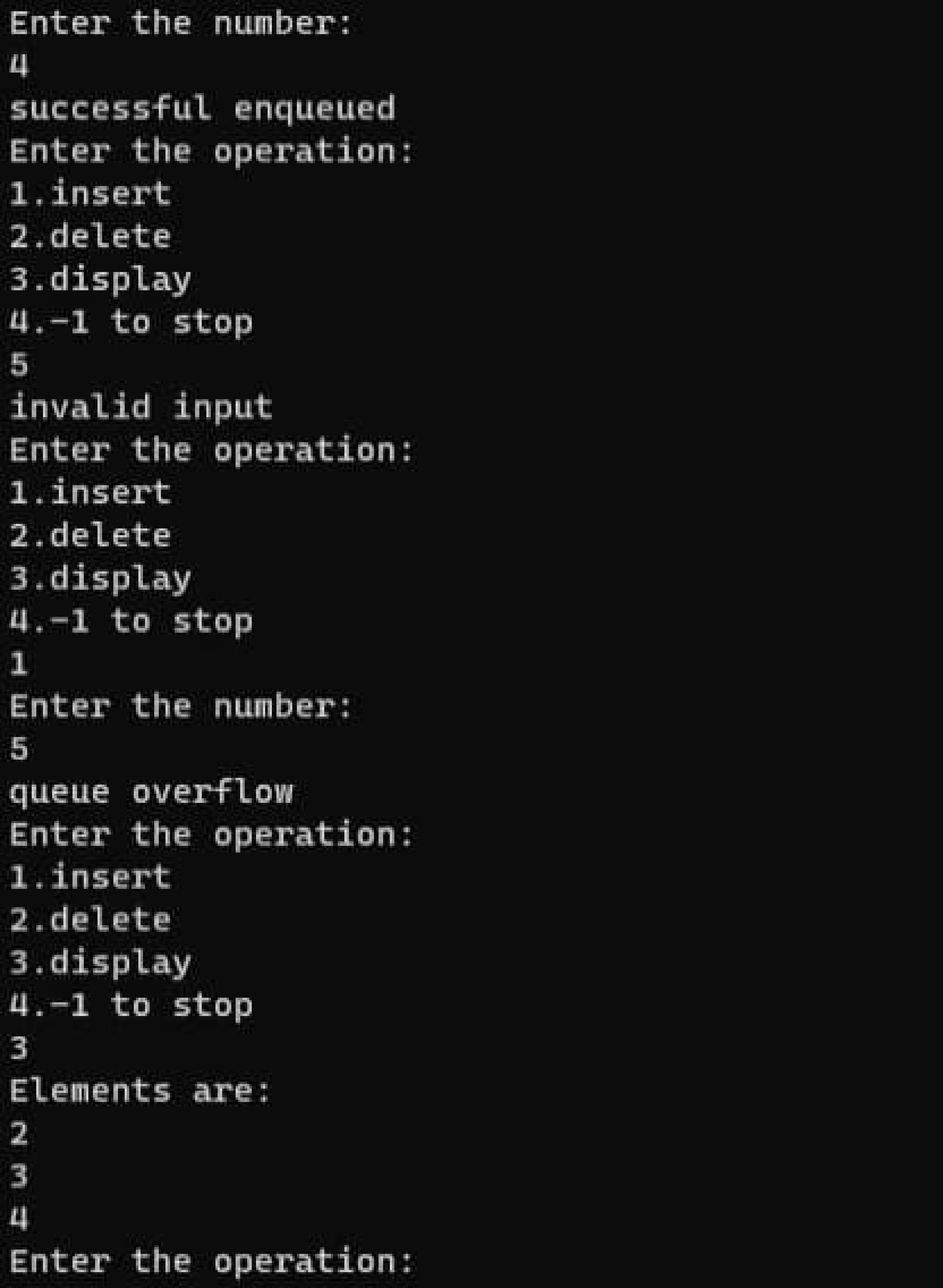
```
Enter the operation
1. Insert
2. Delete
3. Display
4. Exit
1
Enter the number
5
Successfully enqueued.
```



```
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
1
Enter the number:
2
successfully enqueued
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
1
Enter the number:
3
successful enqueued
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
1
Enter the number:
4
successful enqueued
Enter the operation:
1.insert
2.delete
```



```
4.-1 to stop
2
deleted element is=2
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
2
deleted element is=3
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
2
deleted element is=4
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
2
queue underflow
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
```



```
Enter the number:
4
successful enqueued
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
5
invalid input
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
1
Enter the number:
5
queue overflow
Enter the operation:
1.insert
2.delete
3.display
4.-1 to stop
3
Elements are:
2
3
4
Enter the operation:
```

Output :-

Enter the operation

1. Insert
2. Delete
3. Display
4. -1 to stop

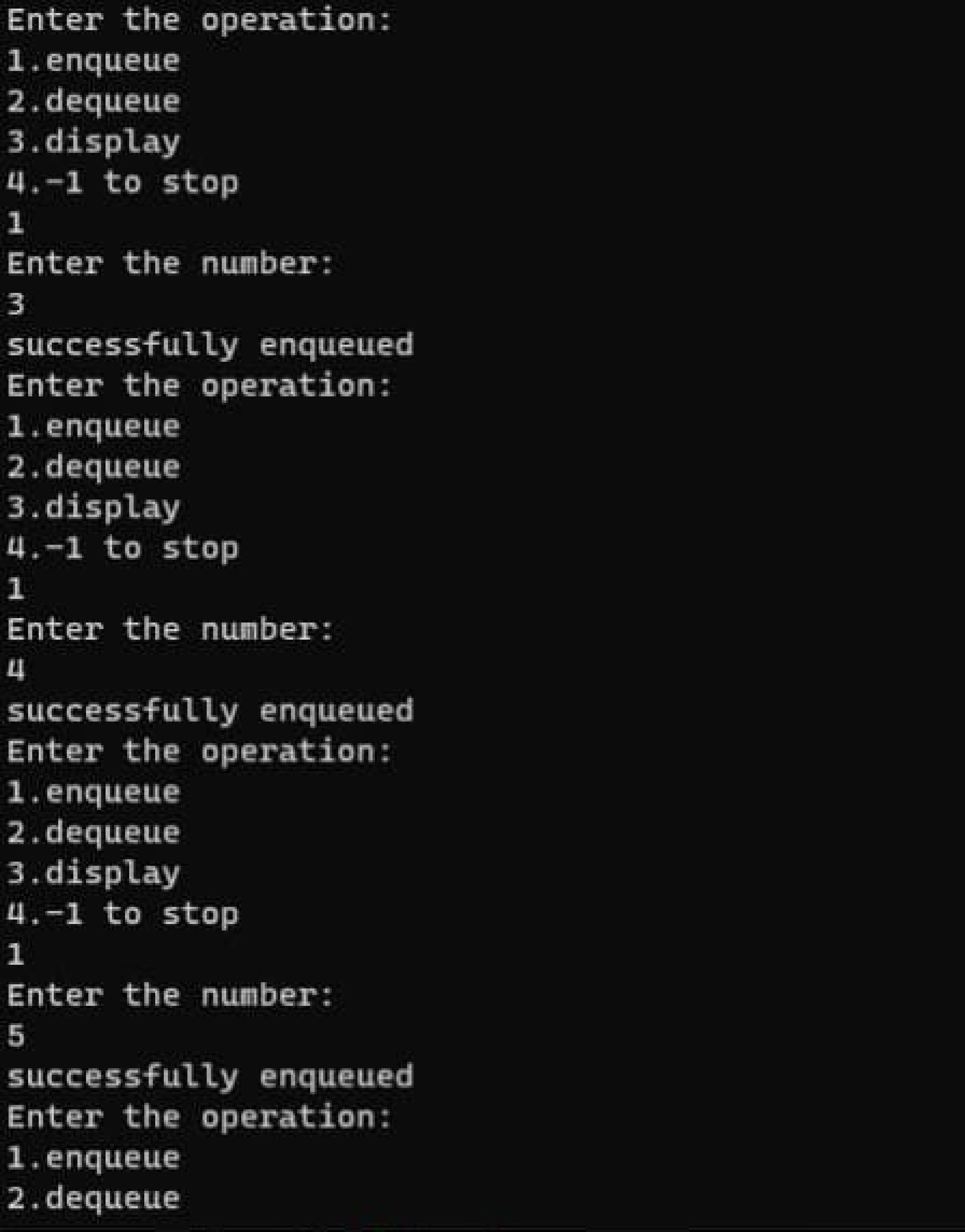
```
1          Enter the number: 5
```

4. Write a C-program to implement CircularQueue (insert, delete, display).

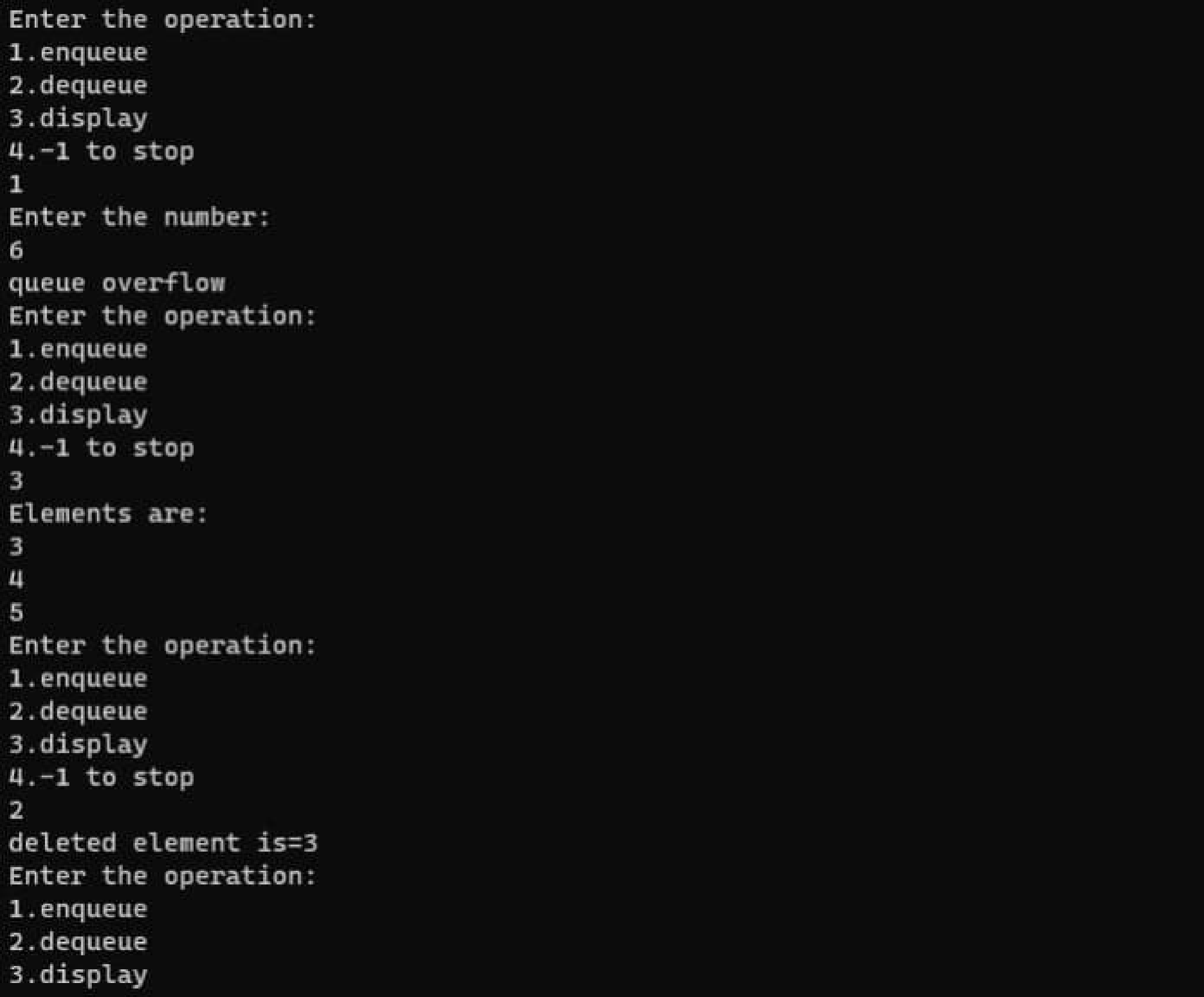
```
# include <stdio.h>
# include <conio.h>
# include <process.h>
 # define QUE_SIZE 5
int item, front =0 , rear=-1, q[QUE_SIZE], count=0;
 void insert()
{
 if (count== QUE_SIZE)
 {
  ~~if~~ printf ("Queue overflow\n");
 return;
 }
 rear=(rear+1)% QUE_SIZE;
 q[rear] = item;
 count++;
 }
 int delete()
 {
  if (count==0) return -1;
  item = q(front);
  front = (front+1)% QUE_SIZE;
  count= count-1;
   return item;
  }
```

```
void displayQ()
{
 int i, f;
 if (count==0)
 {
 printf("Queue is empty\n");
 ~~for(i=1;i<=count;i++)~~
 return;
 }
 f = front;
 printf("contents of Queue\n");
 for (i=1; i<=count; i++)
 {
  printf("%d\n", q[f]);
  f=(f+1)%QUE_SIZE;
  }
  }
  void main()
  {
   int choice;
     {
     printf("\n 1: insert \n 2: delete \n3: display \n4: exit\n");
     printf("Enter the choice \n");
       scanf("%d, &choice);

     switch (choice)
      {
      case 1: printf("Enter the item to be inserted \n");
              scanf("%d", &item);
              insertrear();
              break;
     case 2; item= delete();
             if(item ==-1)
             printf(" Queue is empty \n");
```



```
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
1
Enter the number:
3
successfully enqueued
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
1
Enter the number:
4
successfully enqueued
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
1
Enter the number:
5
successfully enqueued
Enter the operation:
1.enqueue
2.dequeue
```



```
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
1
Enter the number:
6
queue overflow
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
3
Elements are:
3
4
5
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
2
deleted element is=3
Enter the operation:
1.enqueue
2.dequeue
3.display
```