Linear Queue :

```
#include<stdio.h>
#include<stdlib.h>
#define size 3
int queue[size];
int front=-1; int rear=-1;
void insert();
void delete();
void display();
void main(){
   int choice;
   while(1){
      printf("\n Enter choice \n 1.Insert \n 2.Delete \n 3.Display \n 4.Exit\n");
      scanf("%d",&choice);
      switch(choice){
         case 1:insert();
            break;
         case 2:delete();
            break;
         case 3:display();
            break;
         case 4:exit(0);
         default:printf("Invalid choice:");
      }
   }
   return 0;
}
void insert(){
   int val;
   if(rear==size-1||rear==front-1){
      printf("Stack overflow");
   }
   else{
      printf("Enter a value:");
      scanf("%d",&val);
      if(front==-1&&rear==-1){
         front=rear=0;
      }
      else{
         rear+=1;
      }
   }
```

```
    queue[rear]=val;
}
void delete(){
    int val;
    if(front==-1&&rear==-1){
        printf("Queue is empty");
    }
    printf("deleted element %d",queue[front]);
    val=queue[front];
    if(front==rear){
        front=rear=-1;
    }
    else{
        front+=1;
    }
}
void display(){
    if(front==-1&&rear==-1){
        printf("\nQueue is empty");
    }
    for(int i=front;i<=rear;i++){
        printf("%d\t",queue[i]);
    }
}
```

Output

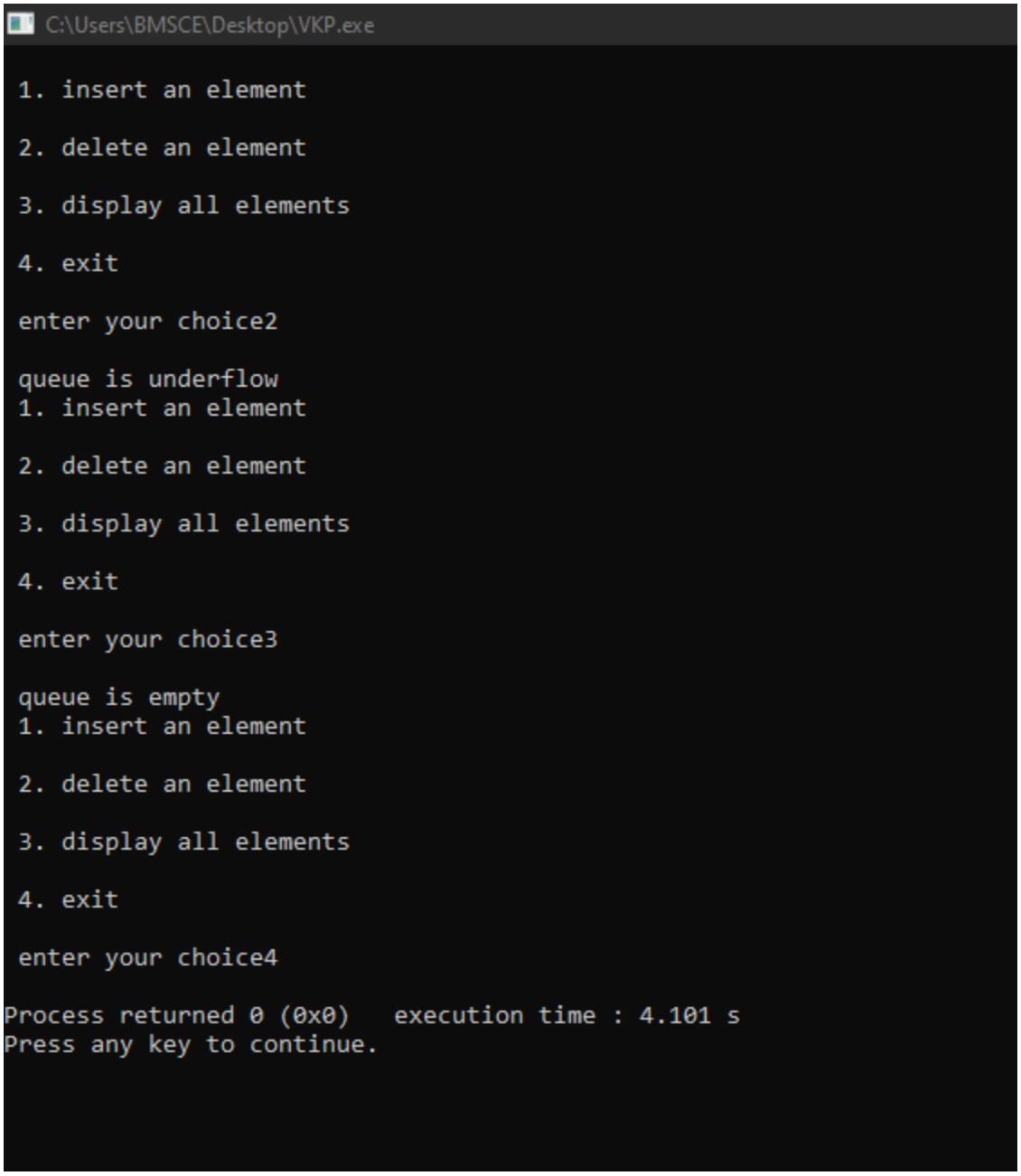
```
/tmp/2zuExQieua.o
Enter choice
 1.Insert
 2.Delete
 3.Display
 4.Exit
1
Enter a value:22
Enter choice
 1.Insert
 2.Delete
 3.Display
 4.Exit
1
Enter a value:5
Enter choice
 1.Insert
 2.Delete
 3.Display
 4.Exit
3
22  5
 Enter choice
 1.Insert
 2.Delete
 3.Display
 4.Exit
```

Circular Queue:

```
#include<stdio.h>
#include<stdlib.h>
#define max 6
int queue[max];
int front=-1;
int rear=-1;

void enqueue(int element)
{

   if(front==-1 && rear == -1)
   {
      front=0;
      rear=0;
      queue[rear]=element;
   }
   else if((rear+1)%max==front)
   {
      printf("queue is overflow");
   }
   else{
      rear=(rear+1)%max;
      queue[rear]=element;
   }
}

int dequeue()
{

   if((front==-1)&&(rear==-1))
   {

      printf("\n queue is underflow");
   }
   else if(front==rear)
   {
      printf("\n the dequeued element is %d", queue[front]);
   front=-1;
   rear=-1;
   }
```

```
    else{
        printf("\n the dequeued element is %d", queue[front]);
        front=(front+1)%max;
    }
}

void display()
{

    int i=front;
    if(front==-1 && rear==-1)
      {
         printf("\n queue is empty");
      }
       else
          {
          printf("\n elements in a queue are:");
        while(i<=rear)
        {
          printf("%d\n", queue[i]);
          i=(i+1)%max;
        }
        }
}

int main()
{

    int choice=1,x;

    while(1)
    {
        printf("\n 1. insert an element\n");
         printf("\n 2. delete an element\n");
          printf("\n 3. display all elements\n");
           printf("\n 4. exit \n");
            printf("\n enter your choice");
            scanf("%d", &choice);

           switch(choice)
           {

              case 1 :  printf("\n enter element to be inserted\n");
              scanf("%d",&x);
```

```
            enqueue(x);
            break;

            case 2 : dequeue();
            break;

                case 3: display();
                break;
                case 4: exit(0);
                break;
                default : printf("enter a valid choice");
        }
    }
    return(0);
}
```


C:\Users\BMSCE\Desktop\VKP.exe
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice2
queue is underflow
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice3
queue is empty
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice4
Process returned 0 (0x0) execution time : 4.101 s
Press any key to continue.