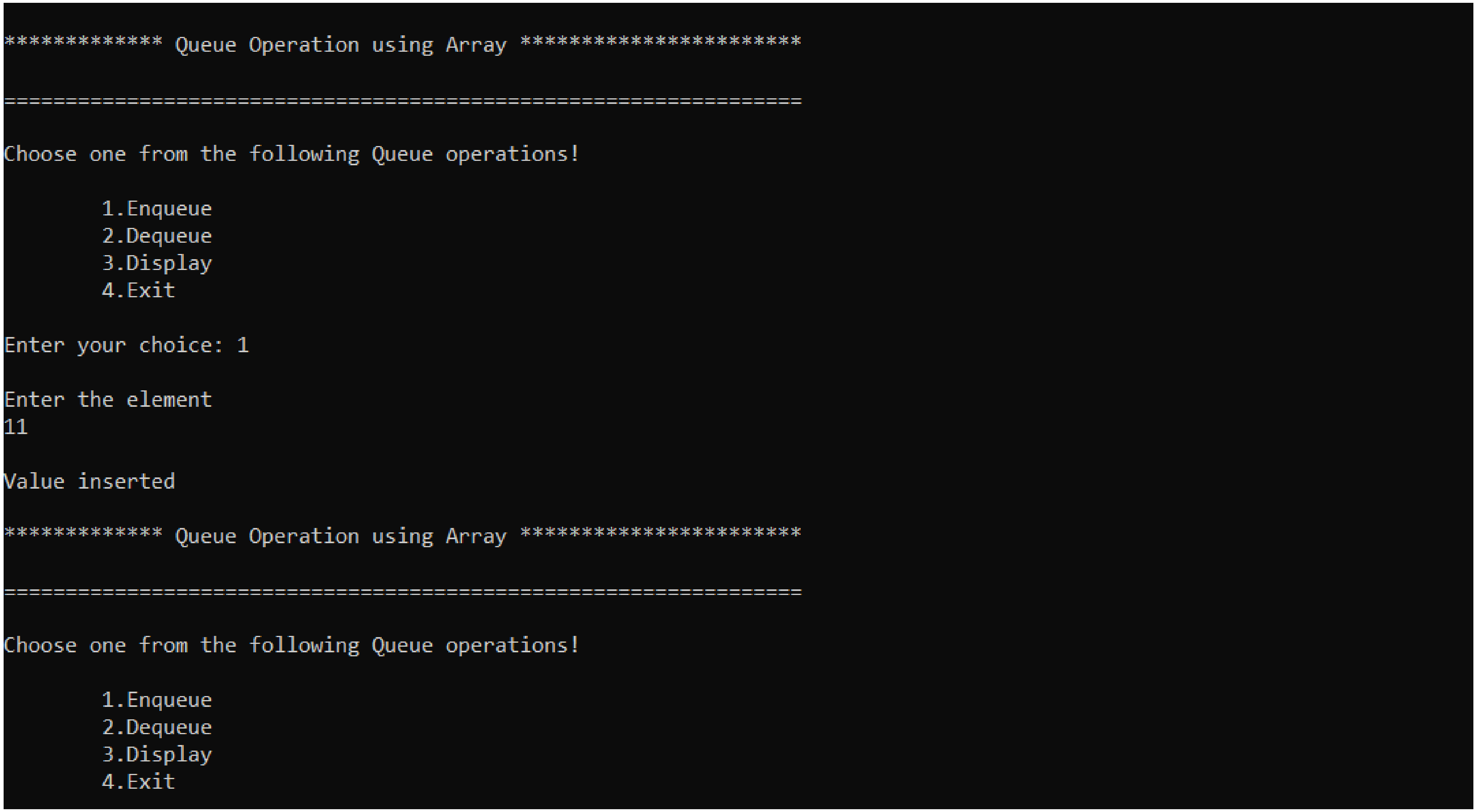
Queue implementation using Array



```
************* Queue Operation using Array ***********************

=================================================================

Choose one from the following Queue operations!

        1.Enqueue
        2.Dequeue
        3.Display
        4.Exit

Enter your choice: 1

Enter the element
11

Value inserted

************* Queue Operation using Array ***********************

=================================================================

Choose one from the following Queue operations!

        1.Enqueue
        2.Dequeue
        3.Display
        4.Exit
```

```
#include<iostream>
#include<stdlib.h>
using namespace std;

#define maxsize 10

//Function Declaration
void enqueue();
void dequeue();
void peek();

//Public variable declaration
int front = -1, rear = -1;
int queue[maxsize];
```

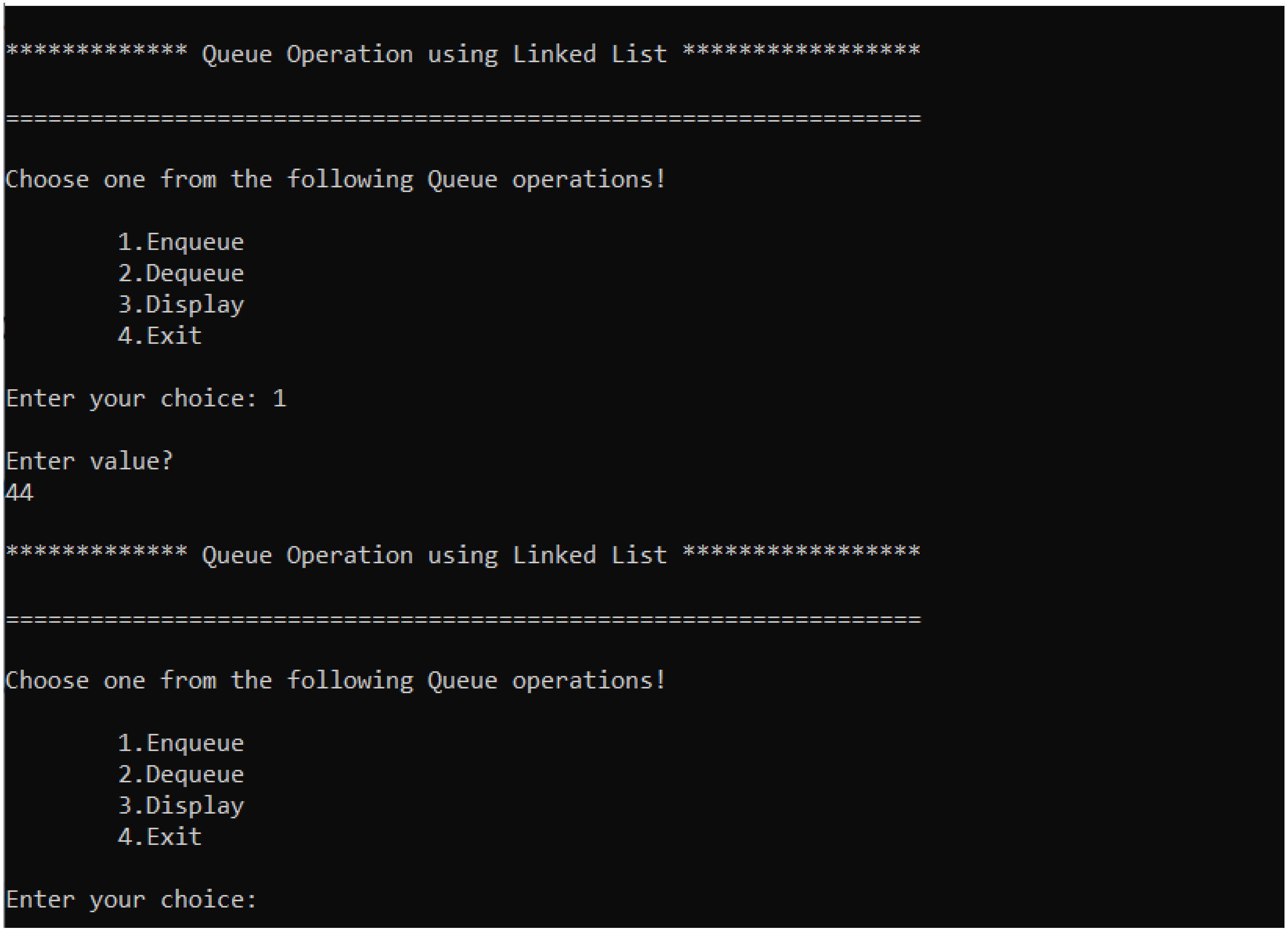
```
//Main Function definition
int main()
{
int choice;
while(choice != 4)   {
cout<<"\n************* Queue Operation using Array
***********************\n";
cout<<"\n=========================================
==========================\n";
cout<<"\nChoose one from the following Queue
operations!\n";
cout<<"\n\t1.Enqueue\n\t2.Dequeue\n\t3.Display\n\t4.
Exit\n";
cout<<"\nEnter your choice: ";
cin>>choice;
switch(choice)
     {
        case 1:
          enqueue();
          break;
        case 2:
          dequeue();
          break;
        case 3:
          peek();
          break;
        case 4:
          exit(0);
          break;
        default:
          cout<<"\nEnter valid choice??\n";
     }  //end of switch statement
   }  //end of while loop
   return 0;
} //end of main function
```

```
//Enqueue function definition
void enqueue()
{
   int item;
   cout<<"\nEnter the element\n";
   cin>>item;
   if(rear == maxsize-1)
   {
      cout<<"\nOVERFLOW\n";
      return;
   }
   if(front == -1 && rear == -1)
   {
      front = 0;
      rear = 0;
   }
   else
   {
      rear = rear+1;
   }
   queue[rear] = item;
   cout<<"\nValue inserted \n";
}
```

```
//Dequeue function definition
void dequeue()
{
   int item;
   if (front == -1 || front > rear)
   {
      cout<<"\nUNDERFLOW\n";
      return;
   }
   else
   {
      item = queue[front];
      if(front == rear)
      {
         front = -1;
         rear = -1 ;
      }
      else
      {
         front = front + 1;
      }
      cout<<"\nvalue deleted \n";
   }
}
```

```
//Peek function definition
void peek()
{
    int i;
    if(rear == -1)
    {
        cout<<"\nEmpty queue\n";
    }
    else
    {   cout<<"\nprinting values .....\n";
        for(i=front;i<=rear;i++)
        {
            cout<<queue[i]<<" ";
        }
    }
    cout<<endl;
}
```

Queue implementation using Linked List



```
************* Queue Operation using Linked List *****************

=================================================================

Choose one from the following Queue operations!

        1.Enqueue
        2.Dequeue
        3.Display
        4.Exit

Enter your choice: 1

Enter value?
44

************* Queue Operation using Linked List *****************

=================================================================

Choose one from the following Queue operations!

        1.Enqueue
        2.Dequeue
        3.Display
        4.Exit

Enter your choice:
```

```
#include<iostream>
#include<stdlib.h>
using namespace std;

//Node creation
struct node
{
   int data;
   struct node *next;
};

//Pointer declaration
struct node *front;
struct node *rear;

//Function declaration
void enqueue();
void dequeue();
void peek();
```

```
//Main Function definition
int main()
{
int choice;
while(choice != 4)   {
cout<<"\n************* Queue Operation using Linked
List *****************\n";
cout<<"\n=======================================
==========================\n";
cout<<"\nChoose one from the following Queue
operations!\n";
cout<<"\n\t1.Enqueue\n\t2.Dequeue\n\t3.Display\n\t4.
Exit\n";
cout<<"\nEnter your choice: ";
cin>>choice;
switch(choice)
      {
         case 1:
           enqueue();
           break;
         case 2:
           dequeue();
           break;
         case 3:
           peek();
           break;
         case 4:
           exit(0);
           break;
         default:
           cout<<"\nEnter valid choice??\n";
      } //end of switch statement
   } //end of while loop
   return 0;
} //end of main function
```

```
//Enqueue function definition
void enqueue()  {
   struct node *ptr;
   int item;
   ptr = (struct node *) malloc (sizeof(struct node));
   if(ptr == NULL)   {
      cout<<"\nOVERFLOW\n";
      return;
   }
   else   {
      cout<<"\nEnter value?\n";
      cin>>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;
      }
   }
}
```

```
//Dequeue function definition
void dequeue ()
{
   struct node *ptr;
   if(front == NULL)
   {
      cout<<"\nUNDERFLOW\n";
      return;
   }
   else
   {
      ptr = front;
      front = front -> next;
      delete ptr;
   }
}
```

```
//Peek function definition
void peek()
{
    struct node *ptr;
    ptr = front;
    if(front == NULL)
    {
        cout<<"\nEmpty queue\n";
    }
    else
    {   cout<<"\nprinting values .....\n";
        while(ptr != NULL)
        {
            cout<<ptr -> data<<" ";
            ptr = ptr -> next;
        }
    }
}
```