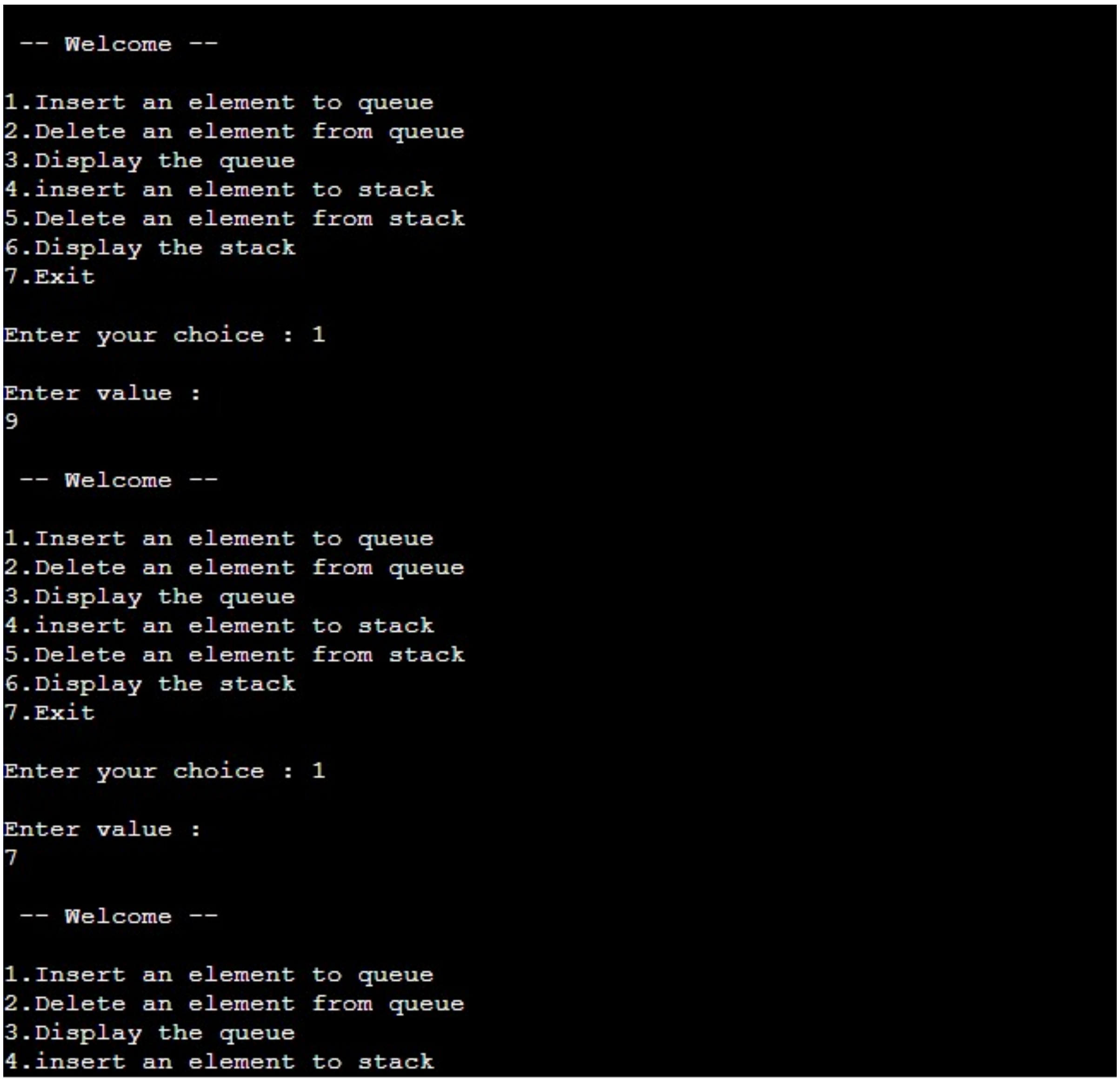
PROGRAM 8

Write a program to implement stacks and queues using Linked List Representation -

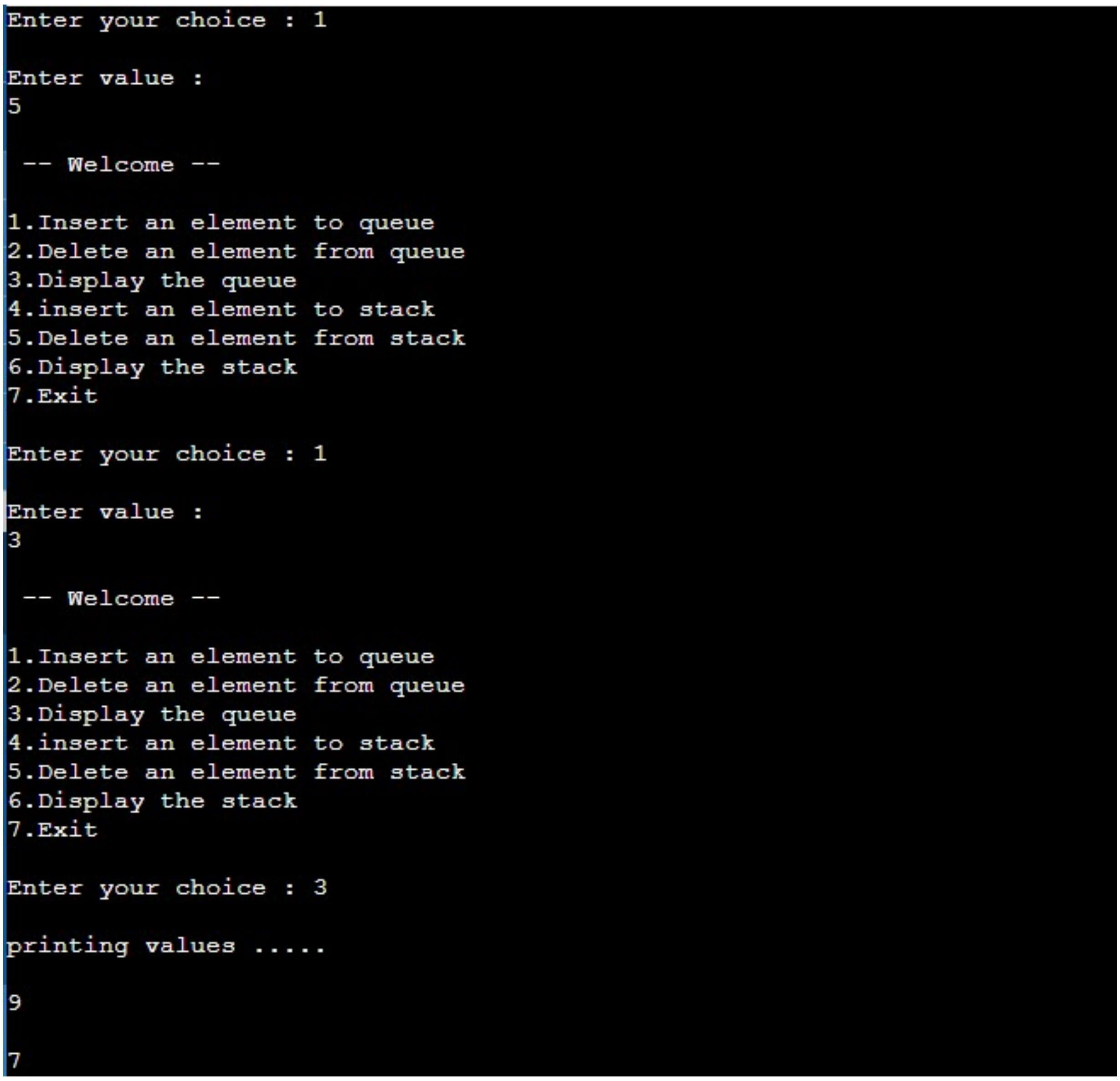
```
#include<stdio.h>
#include<stdlib.h>
struct nodeq
{
    int data;
    struct nodeq *nextq;
};
struct nodeq *frontq;
struct nodeq *rearq;
void insertq();
void deleteq();
void displayq();
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

int main ()
{
    int choice;
    while(choice != 7)
    {
        printf("\n*************************Main 
Menu*****************************\n");

printf("\n===============================================
==================\n");
        printf("\n1.insert an element to queue\n2.Delete 
an element from queue\n3.Display the queue\n4.insert an 
element to stack\n5.Delete an element from 
stack\n6.Display the stack\n7.Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            insertq();
            break;
            case 2:
            deleteq();
            break;
            case 3:
            displayq();
            break;
            case 4:
                push();
                break;
            case 5:
                pop();
                break;
            case 6:
                display();
                break;
            case 7:
            return(0);
            break;
            default:
            printf("\nEnter valid choice??\n");
            return 0;
        }
    }
}
void insertq()
{
    struct nodeq *ptr;
    int item;

    ptr = (struct nodeq *) malloc (sizeof(struct nodeq));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
        return;
    }
    else
    {
        printf("\nEnter value?\n");
        scanf("%d",&item);
        ptr -> data = item;
        if(frontq == NULL)
        {
            frontq = ptr;
            rearq = ptr;
            frontq -> nextq = NULL;
            rearq -> nextq = NULL;
        }
        else
        {
            rearq -> nextq = ptr;
            rearq = ptr;
            rearq->nextq = NULL;
        }
    }
}
void deleteq ()
{
    struct nodeq *ptr;
    if(frontq == NULL)
    {
        printf("\nUNDERFLOW\n");
        return;
    }
    else
    {
        ptr = frontq;
        frontq = frontq -> nextq;
        free(ptr);
    }
}
```

```
void displayq()
{
    struct nodeq *ptr;
    ptr = frontq;
    if(frontq == NULL)
    {
        printf("\nEmpty queue\n");
    }
    else
    {   printf("\nprinting values .....\n");
        while(ptr != NULL)
        {
            printf("\n%d\n",ptr -> data);
            ptr = ptr -> nextq;
        }
    }
}

void push ()
{
    int val;
    struct node *ptr = (struct node*)malloc(sizeof(struct
node));
    if(ptr == NULL)
    {
        printf("not able to push the element");
    }
    else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(head==NULL)
        {
            ptr->val = val;
            ptr -> next = NULL;
            head=ptr;
        }
        else
        {
            ptr->val = val;
            ptr->next = head;
            head=ptr;

        }
        printf("Item pushed");

    }
}

void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");

    }
}
void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
```

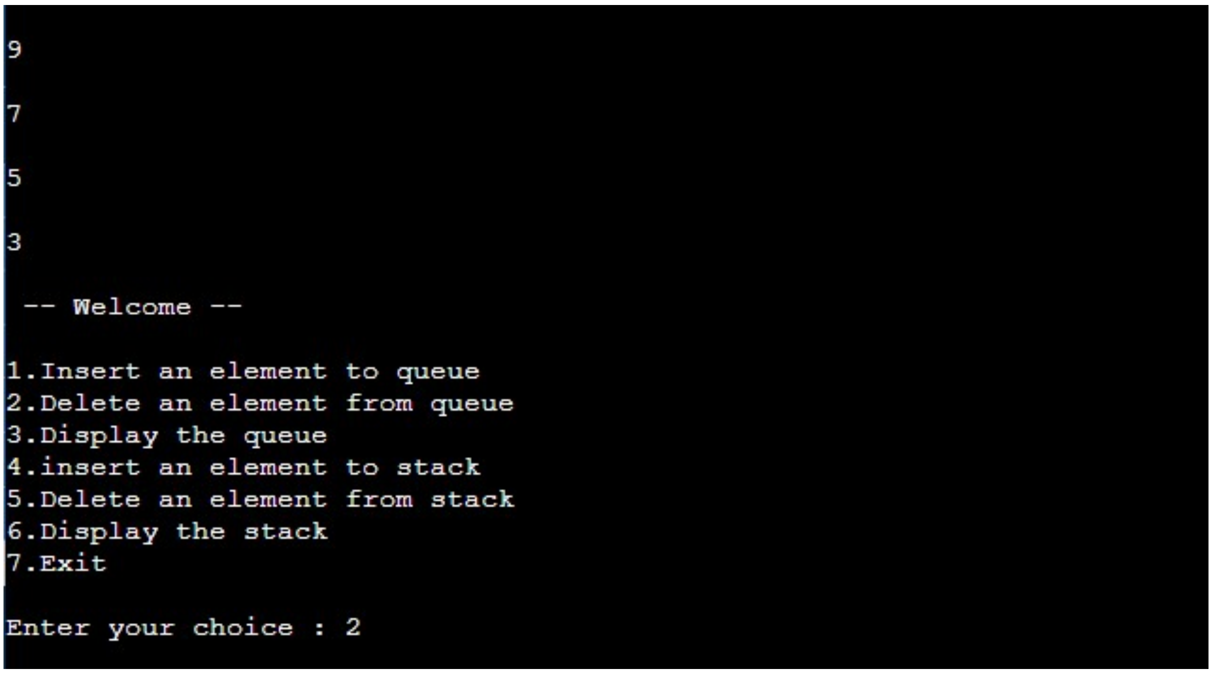
Output –



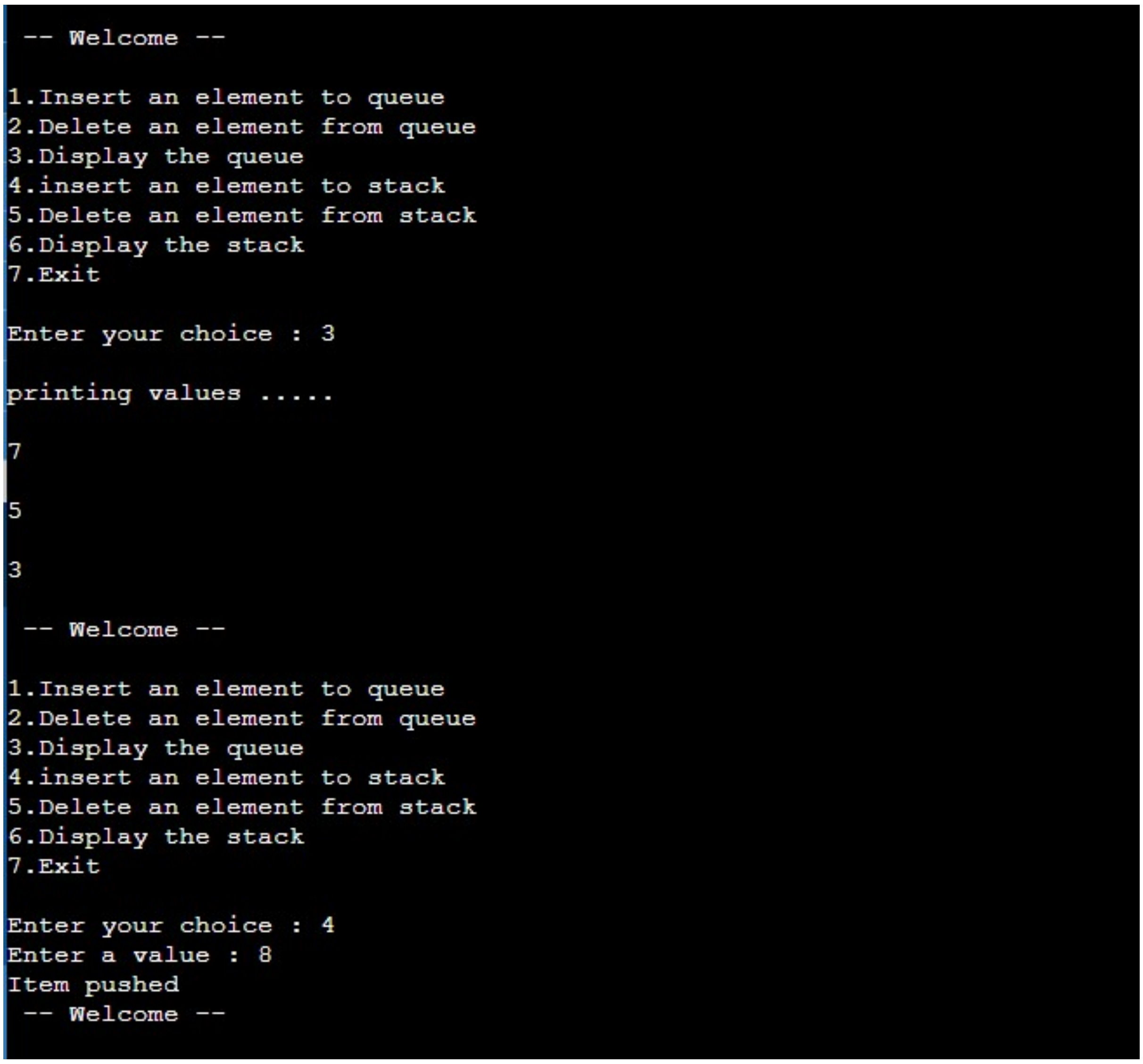
```
 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 1

Enter value :
9

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 1

Enter value :
7

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
```



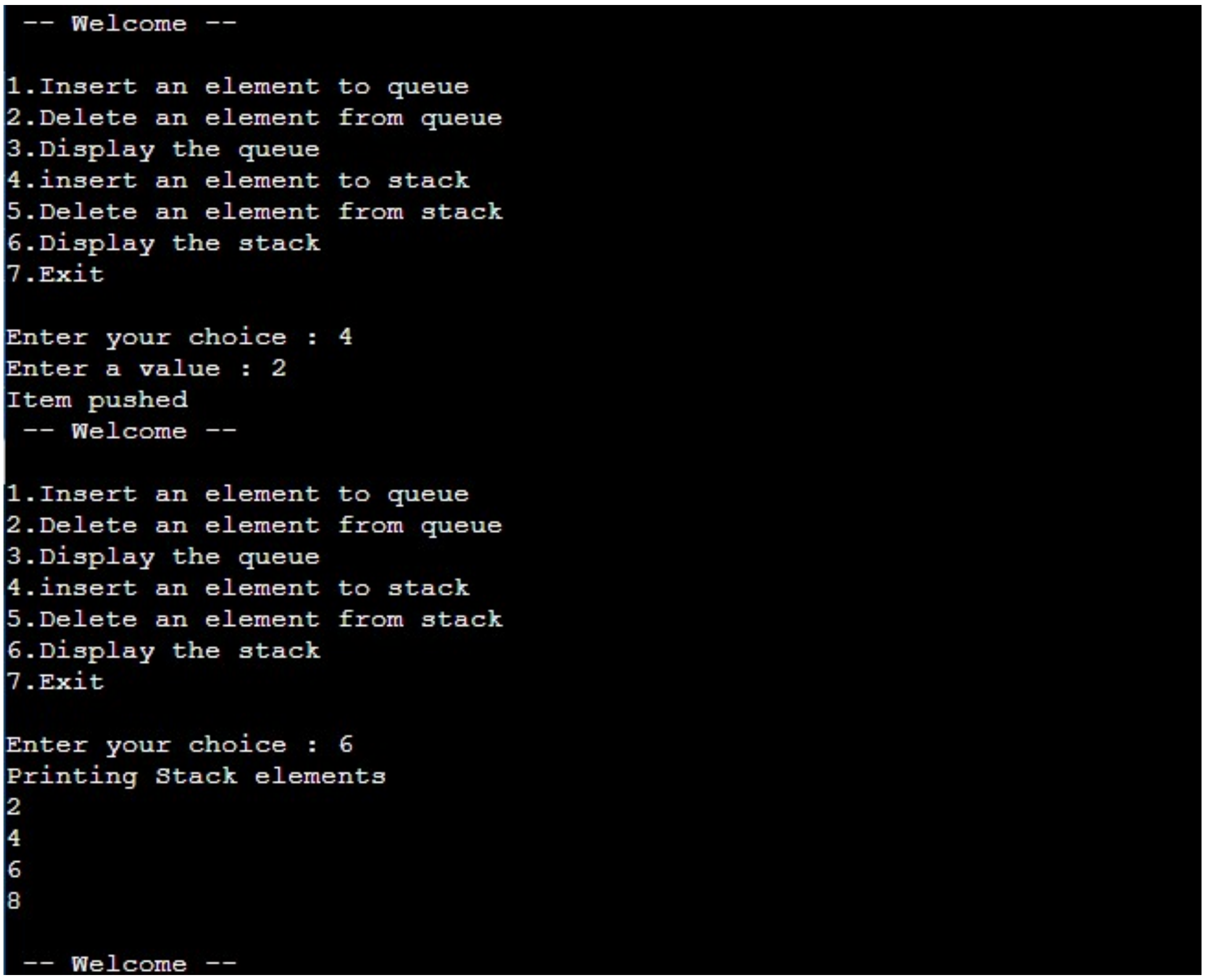
```
Enter your choice : 1

Enter value :
5

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 1

Enter value :
3

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 3

printing values .....

9

7
```



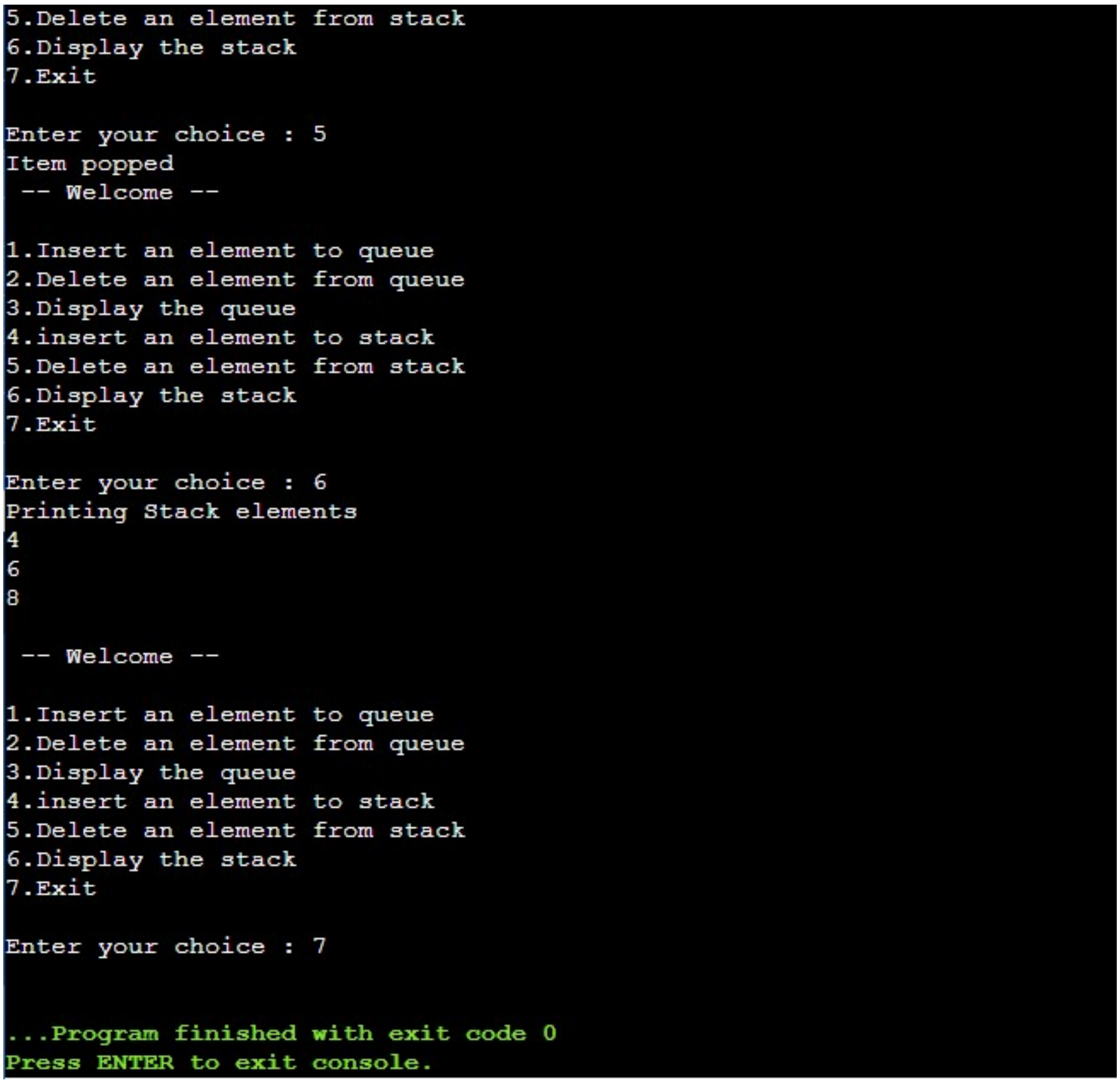
```
9

7

5

3

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 2
```



```
 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 3

printing values .....

7

5

3

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 4
Enter a value : 8
Item pushed
 -- Welcome --
```



```
 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 4
Enter a value : 2
Item pushed
 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 6
Printing Stack elements
2
4
6
8

 -- Welcome --
```



```
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 5
Item popped
 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 6
Printing Stack elements
4
6
8

 -- Welcome --

1.Insert an element to queue
2.Delete an element from queue
3.Display the queue
4.insert an element to stack
5.Delete an element from stack
6.Display the stack
7.Exit

Enter your choice : 7


...Program finished with exit code 0
Press ENTER to exit console.
```