Experiment No.4
Implementation of Queue menu driven program using arrays
Name: AARYAN CHANDRAKANT GOLE
Roll No: 12
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No. 4: Simple Queue Operations

Aim: To implement a Linear Queue using arrays.

Objective:

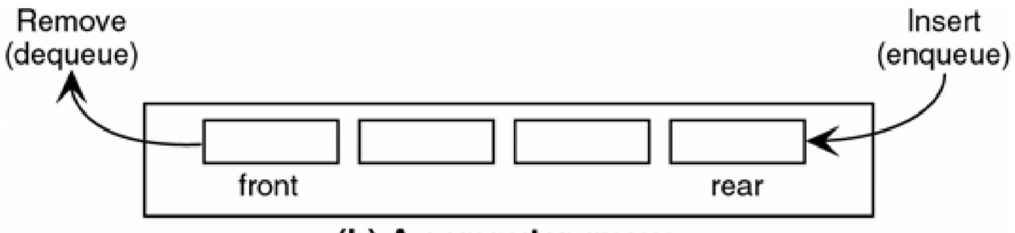
1 Understand the Queue data structure and its basic operations.

2. Understand the method of defining Queue ADT and its basic operations.

3. Learn how to create objects from an ADT and member functions are invoked.

Theory:

A queue is an ordered collection where items are removed from the front and inserted at the rear, following the First-In-First-Out (FIFO) order. The fundamental operations for a queue are "Enqueue," which adds an item to the rear, and "Dequeue," which removes an item from the front.



(b) A computer queue

Typically, a one-dimensional array is used to implement a queue, and two integer values, FRONT and REAR, track the front and rear positions in the array. When an element is removed from the queue, FRONT is incremented by one, and when an element is added to the queue, REAR is increased by one. This ensures that items are processed in the order they were added, maintaining the FIFO principle.

Algorithm:

```
ENQUEUE(item)

1.  If (queue is full)
      Print "overflow"
2.  if (First node insertion)
       Front++
3.  rear++
Queue[rear]=value
```

```
DEQUEUE()
1.  If (queue is empty)
        Print “underflow”
2.  if(front=rear)
        Front=-1 and rear=-1
3.  t = queue[front]
4.  front++
5.  Return t
```

```
ISEMPTY()
1. If(front = -1)then
       return 1
2. return 0
```

```
ISFULL()
1. If(rear = max)then
       return 1
2. return 0
```

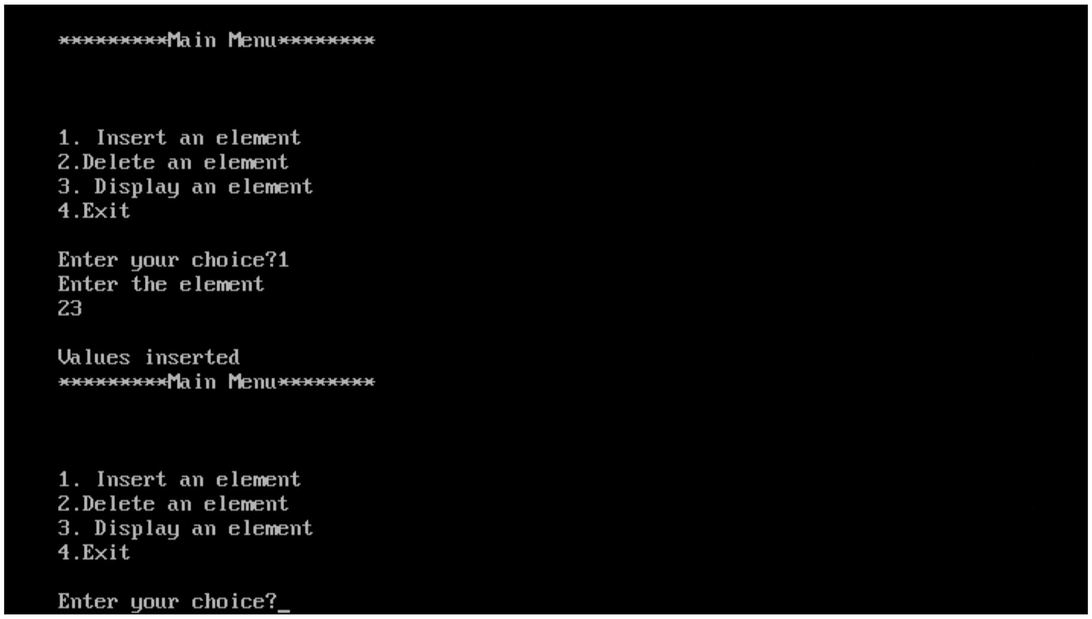
Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define maxsize 5

void insert();
void deleted();
void display();
int front=-1,rear=-1;
int queue[maxsize];

void main()
{
        int choice;
        clrscr();
        while(choice!=4)
        {
                printf("\n*********Main Menu********\n");
                printf("\n                                        \n");
                printf("\n1. Insert an element\n2.Delete an element\n3. Display an
element\n4.Exit\n") ;
                printf("\nEnter your choice?");
                scanf("%d",&choice);
                switch(choice)
                {
                    case 1:
                    insert();
                    break;
                    case 2:
                    deleted();
                    break;
                    case 3:
                    display();
                    break;
                    case 4:
                    exit(0);
                    break;
                    default:
                    printf("\nEnter valid choice??\n");
                }
        }
        getch();
}
void insert()
{
        int item;
        printf("\Enter the element\n");
        scanf("\n%d",&item);
        if(rear==maxsize-1)
        {
                printf("\nOVERFLOW\n");
                return;
```

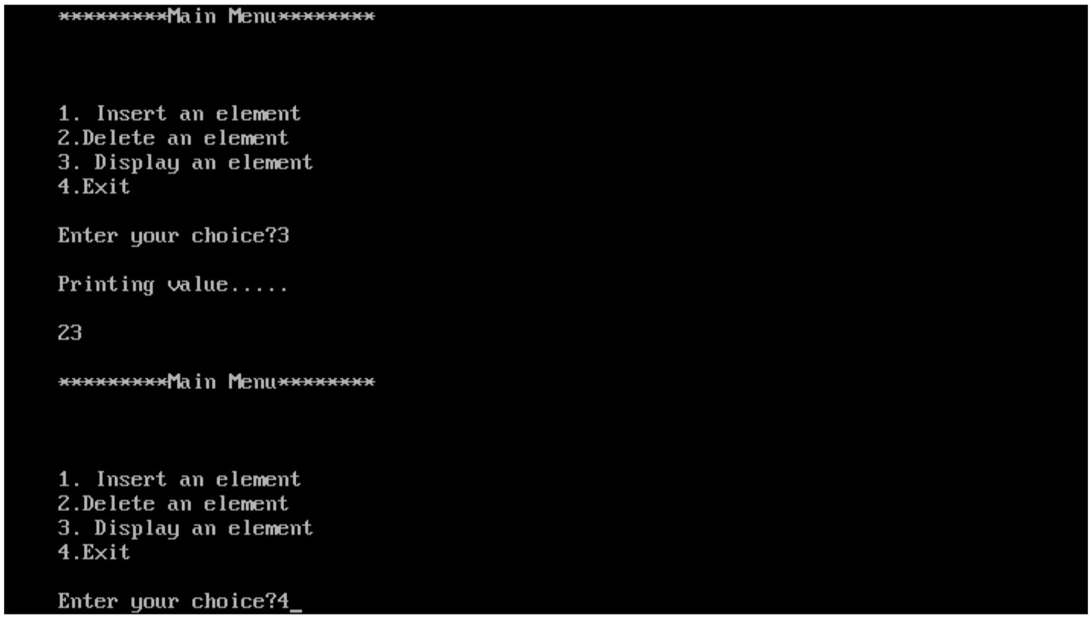
```
          }
          else if(front==-1 && rear==-1)
          {
                    front=0;
                    rear=0;
          }
          else
          {
                    rear=rear+1;
          }
          queue[rear]=item;
          printf("\nValues inserted");
}

void deleted()
{
          int item;
          if(front==-1||front>rear)
          {
                    printf("\nUNDERFLOW\n");
                    return;
          }
          else
          {
                    item=queue[front];

                    if(front==rear)
```

```
		{
			front=-1;
			rear=-1;
		}
		else
		{
			front=front+1;
		}
		printf("\n value deleted");
	}
}

void display()
{
	int i;
	if(rear==-1)
	{
		printf("\nEmpty queue\n");
	}
	else
	{
		printf("\nPrinting value.....\n");
		for(i=front;i<=rear;i++)
		{
			printf("\n%d\n",queue[i]);
		}
	}}
```

Output:



```
*********Main Menu********



1. Insert an element
2.Delete an element
3. Display an element
4.Exit

Enter your choice?1
Enter the element
23

Values inserted
*********Main Menu********



1. Insert an element
2.Delete an element
3. Display an element
4.Exit

Enter your choice?_
```



```
*********Main Menu********



1. Insert an element
2.Delete an element
3. Display an element
4.Exit

Enter your choice?3

Printing value.....

23

*********Main Menu********



1. Insert an element
2.Delete an element
3. Display an element
4.Exit

Enter your choice?4_
```

Conclusion:

1) Elaborate the evaluation of the following postfix expression in your program.

AB+C-

The given program is designed to evaluate postfix expressions. The expression "AB+C-" would be evaluated as follows:

- Push 'A' onto the stack.
- Push 'B' onto the stack.
- When encountering '+', pop 'B' and 'A' from the stack, calculate 'B + A' (which is 'B' + 'A' in this case), and push the result back onto the stack.
- Push 'C' onto the stack.
- When encountering '-', pop 'C' and the result of the previous addition ('B' + 'A') from the stack, calculate 'C - (B + A)', and push the result back onto the stack.

After evaluating the entire expression, the program will print the result, which should be the answer to the expression "AB+C-".

2) Will this input be accepted by your program. If so, what is the output?

The input "AB+C-" should be accepted by the program. The output will depend on the values of 'A', 'B', and 'C'. The program will read these values and evaluate the expression accordingly. You need to provide actual values for 'A', 'B', and 'C' during runtime for the program to calculate and display the result.