Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: 3rd - Fall semester / Year:2022

BSc in CSE (Day)

LAB REPORT NO - 04

Course Title: Data Structure Lab

Course Code: CSE 106 / Section: DE-221

Lab Experiment Tittle : Queue Operations.

Student Details

	Name	ID
1.	Khondokar Saim	221902353

Lab Date : 30 / 11 / 2022

Submission Date : 08 / 12 / 2022

Course Teacher's Name : Farhana Akter Sunny.

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: .... Signature: ....

Comments: .... Date: ....

❑ Tittle of the Lab Experiment :

Implement a C program for Queue Operations

❑ Objectives :

- A queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.

- A queue is an object (an abstract data structure - ADT) that allows the following operations:

➢ Enqueue : Add an element to the end of the queue
➢ Dequeue : Remove an element from the front of the queue
➢ IsEmpty : Check if the queue is empty
➢ IsFull : Check if the queue is full
➢ Peek : Get the value of the front of the queue without removing it

- Real Life example of a queue data structure

➢ People on an escalator
➢ Cashier line in a store
➢ A car wash line
➢ One way exits

- Types of Queues in Data Structure

➢ Simple Queue
➢ Circular Queue
➢ Priority Queue
➢ Double-Ended Queue (Deque)

1. **Implement a C program for array insertion and deletion. Be informed that, you have to use 2 functions one for insertion and one for deletion.**

Algorithm :

```
Step – 1 =   Declear function prototype
             enqueue() for element insert
             dequeue() for element insert for element delete
             getRear() for print rear element
             getFront() for print front  element

Step – 2 =   Create queue array implementation program Menu option

Step – 3 =   Write a code for Switch case

Step – 4 =   enqueue() for element insert

             {
                 if (isFull())
                 {
                     return 0;
                 }

                 rear = (rear + 1) % CAPACITY;
                 size++;

                 queue[rear] = data;

                 return 1;
             }
```

Step – 5 = dequeue() for element insert for element delete

```
{
   int data = INT_MIN;

   if (isEmpty())
   {
      return INT_MIN;
   }

   data = queue[front];

   front = (front + 1) % CAPACITY;

   size--;

   return data;
}
```

Step – 6 = getRear() for print rear element

```
{
   return (isEmpty())
         ? INT_MIN
         : queue[rear];
}
```

Step – 6 = getFront() for print front element

```
{
   return (isEmpty())
         ? INT_MIN
         : queue[front];
}
```

Step – 7 = Call all Created function in main () function into switch case

Code:

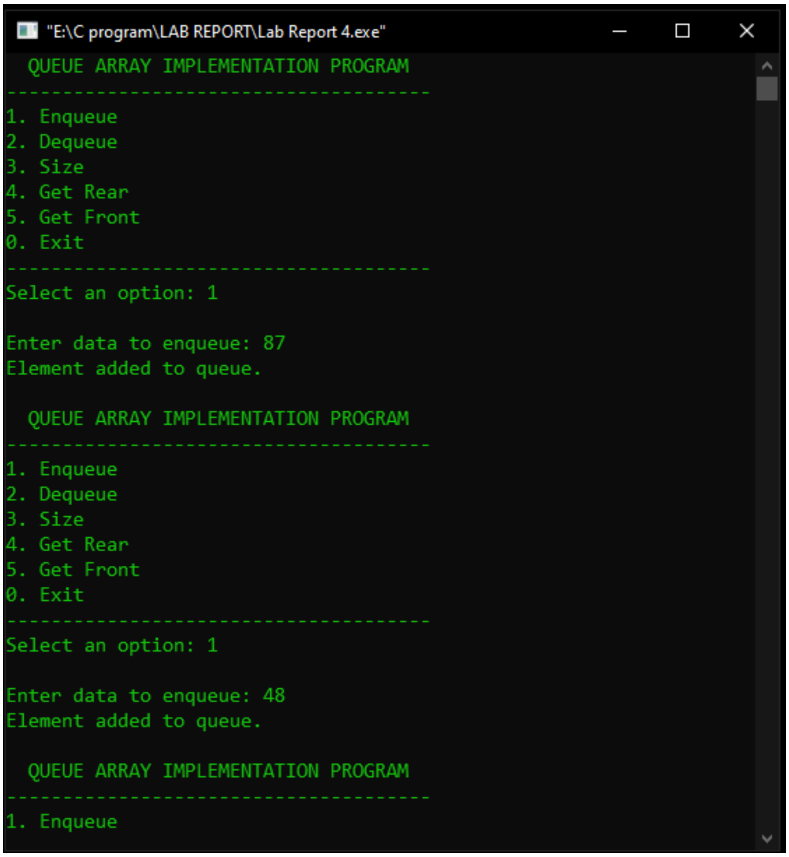
```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define CAPACITY 100

int queue[CAPACITY];
unsigned int size  = 0;
unsigned int rear  = CAPACITY - 1;
unsigned int front = 0;



//Function declaration for various operations on queue
int enqueue(int data);
int dequeue();
int isFull();
int isEmpty();
int getRear();
int getFront();

int main()
{
   int ch, data;

    while (1)
   {
      printf("  QUEUE ARRAY IMPLEMENTATION PROGRAM  \n");
      printf("--------------------------------------\n");
      printf("1. Enqueue\n");
      printf("2. Dequeue\n");
      printf("3. Size\n");
      printf("4. Get Rear\n");
      printf("5. Get Front\n");
      printf("0. Exit\n");
      printf("--------------------------------------\n");
      printf("Select an option: ");

      scanf("%d", &ch);
```

```
switch (ch)
{
    case 1:
        printf("\nEnter data to enqueue: ");
        scanf("%d", &data);

        if (enqueue(data))
            printf("Element added to queue.");
        else
            printf("Queue is full.");

        break;

    case 2:
        data = dequeue();

        if (data == INT_MIN)
            printf("Queue is empty.");
        else
            printf("Data => %d", data);

        break;

    case 3:

        if (isEmpty())
            printf("Queue is empty.");
        else
            printf("Queue size => %d", size);

        break;

    case 4:

        if (isEmpty())
            printf("Queue is empty.");
        else
            printf("Rear => %d", getRear());

        break;
```

```
            case 5:

                if (isEmpty())
                    printf("Queue is empty.");
                else
                    printf("Front => %d", getFront());

                break;

            case 0:
                printf("Exiting from app.\n");
                exit(0);

            default:
                printf("Invalid choice, please input number between (0-5).");
                break;
        }

        printf("\n\n");
    }
}

int enqueue(int data)
{
    if (isFull())
    {
        return 0;
    }

    rear = (rear + 1) % CAPACITY;
    size++;

    queue[rear] = data;

    return 1;
}
```

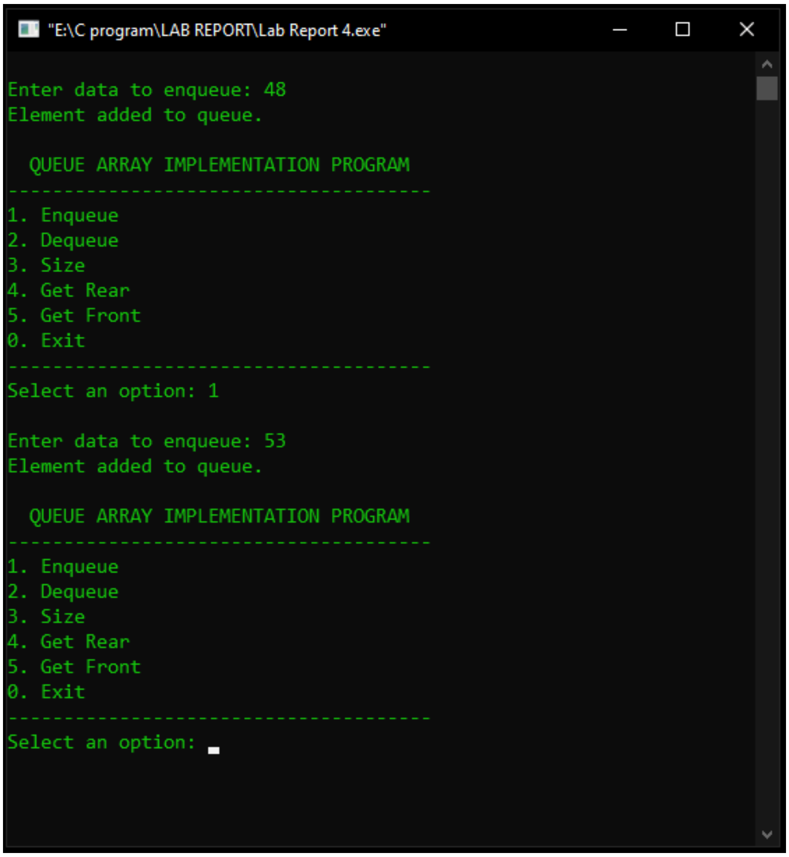
```
int dequeue()
{
    int data = INT_MIN;

    if (isEmpty())
    {
        return INT_MIN;
    }

    data = queue[front];

    front = (front + 1) % CAPACITY;

    size--;

    return data;
}

int isFull()
{
    return (size == CAPACITY);
}

int isEmpty()
{
    return (size == 0);
}

int getFront()
{
    return (isEmpty())
            ? INT_MIN
            : queue[front];
}

int getRear()
{
    return (isEmpty())
            ? INT_MIN
            : queue[rear];
}
```

Output:



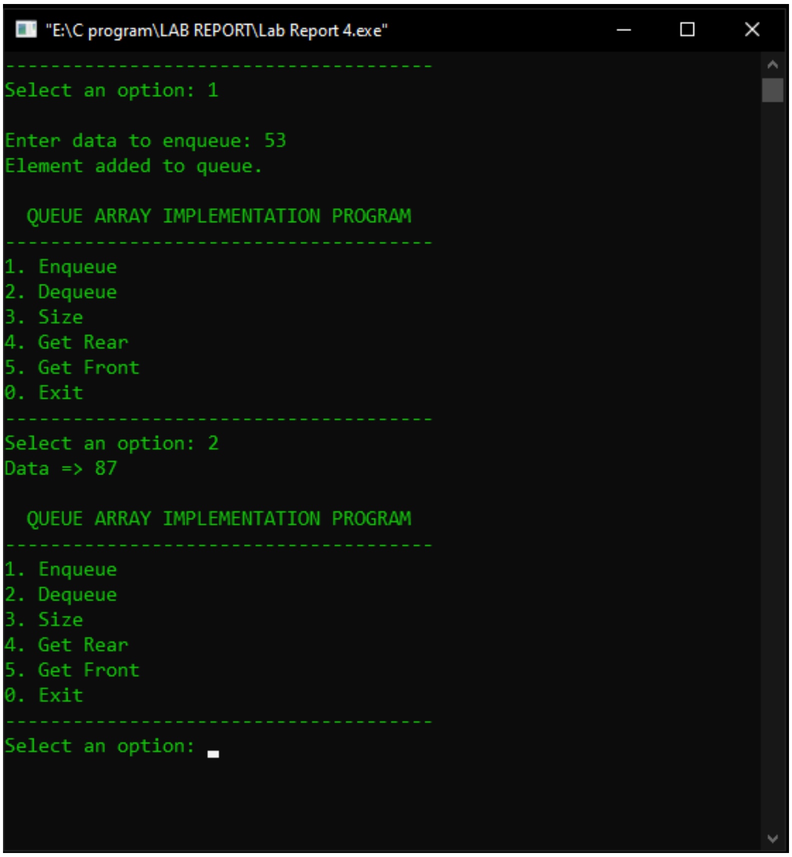
```
"E:\C program\LAB REPORT\Lab Report 4.exe"
  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 1

Enter data to enqueue: 87
Element added to queue.

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 1

Enter data to enqueue: 48
Element added to queue.

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
```

Case – 1 For Element's enqueue



```
"E:\C program\LAB REPORT\Lab Report 4.exe"

Enter data to enqueue: 48
Element added to queue.

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 1

Enter data to enqueue: 53
Element added to queue.

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: _
```

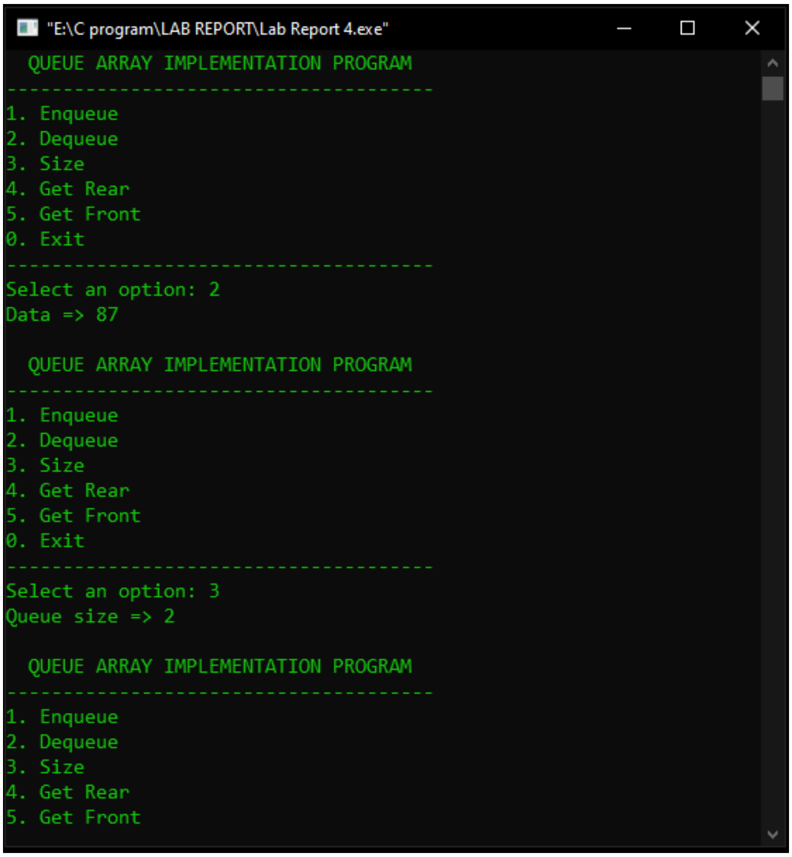
Case – 1.2 For Element's enqueue



"E:\C program\LAB REPORT\Lab Report 4.exe"

```
--------------------------------------
Select an option: 1

Enter data to enqueue: 53
Element added to queue.

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 2
Data => 87

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: _
```

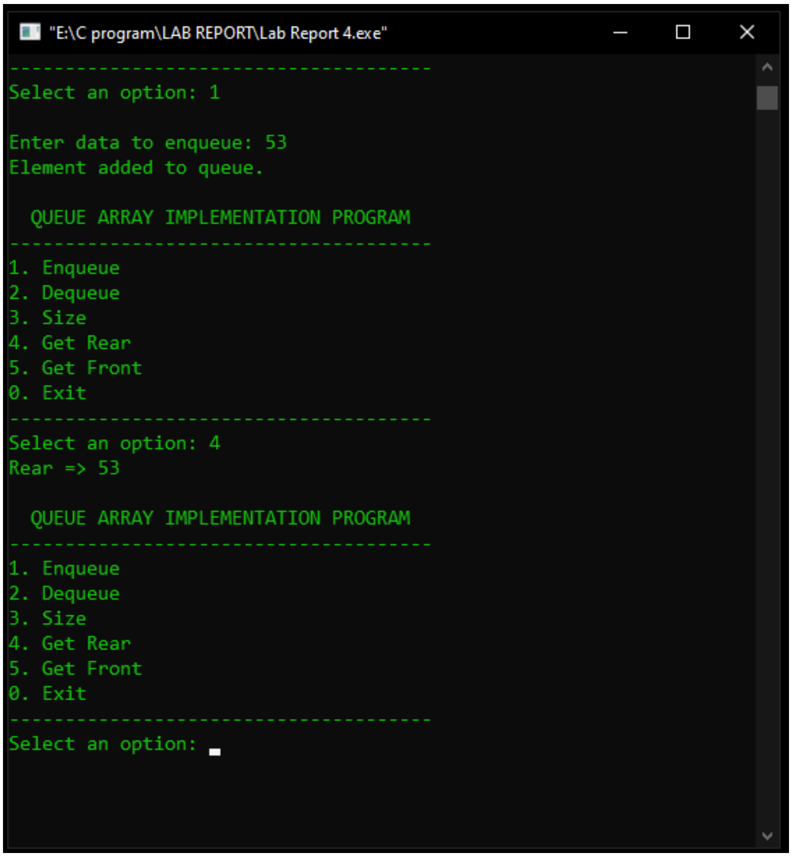
Case –2 For Element’s dequeue



"E:\C program\LAB REPORT\Lab Report 4.exe"

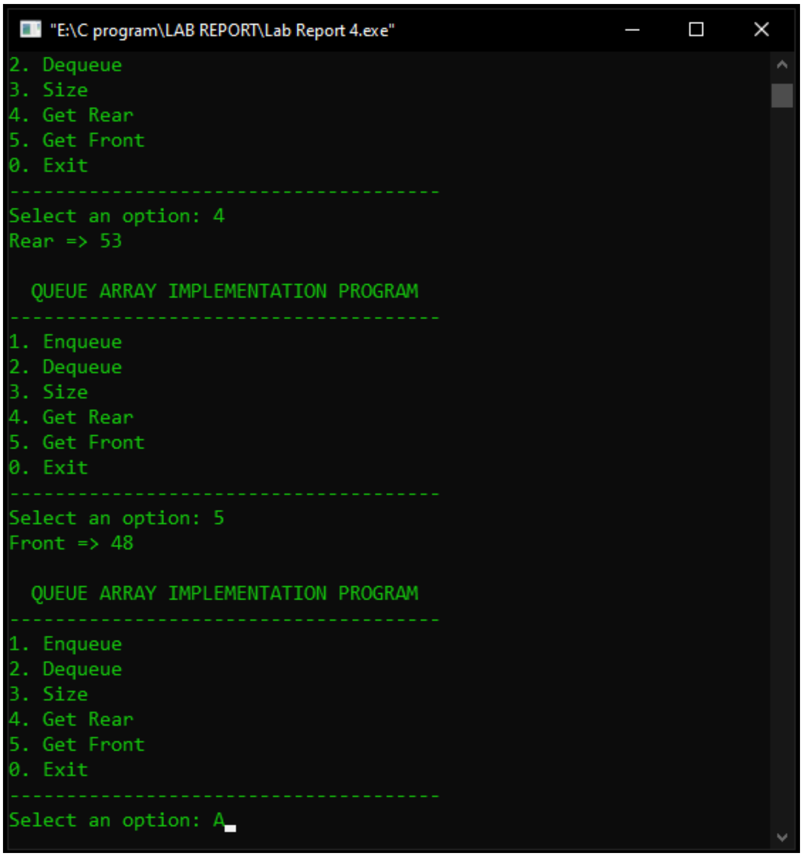
```
  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 2
Data => 87

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 3
Queue size => 2

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
```

Case –3 For Element’s size



```
--------------------------------------
Select an option: 1

Enter data to enqueue: 53
Element added to queue.

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 4
Rear => 53

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: _
```

Case –4 For Element’s Rear



```
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 4
Rear => 53

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: 5
Front => 48

  QUEUE ARRAY IMPLEMENTATION PROGRAM
--------------------------------------
1. Enqueue
2. Dequeue
3. Size
4. Get Rear
5. Get Front
0. Exit
--------------------------------------
Select an option: A_
```

Case –5 For Element’s front

❑ Analysis and Discussion :

- We got the exact result on output. Sometimes the result was wrong but we found the right implementation.
- The problem of displaying anything in output is the easiest implementation. We solve that very easily.
- In this assignment, we faced some problems in this question but with the teacher's help we solve it.
- All program is easy to understand and these helped me a lot to remove my confusion about c programming and queue’s operations.
- I learnt display something in program, Queue operations as like insertion and deletion , switch statement and application of user-defined function etc on program and many basic things about c programming.