Department of
Computer Science and Engineering

Title: Implement Queue using Arrays

Data Structure Lab

CSE 106

Green University of Bangladesh

1 Objective(s)

- To attain knowledge on the Queue data structure and how it works.
- To implement Queue using Arrays.

2 Problem analysis

Queue is also an abstract data type or a linear data structure, just like stack data structure, in which the first element is inserted from one end called the REAR (also called tail), and the removal of existing element takes place from the other end called as FRONT (also called head). This makes queue as FIFO (First In First Out) data structure, which means the element inserted first will be removed first.

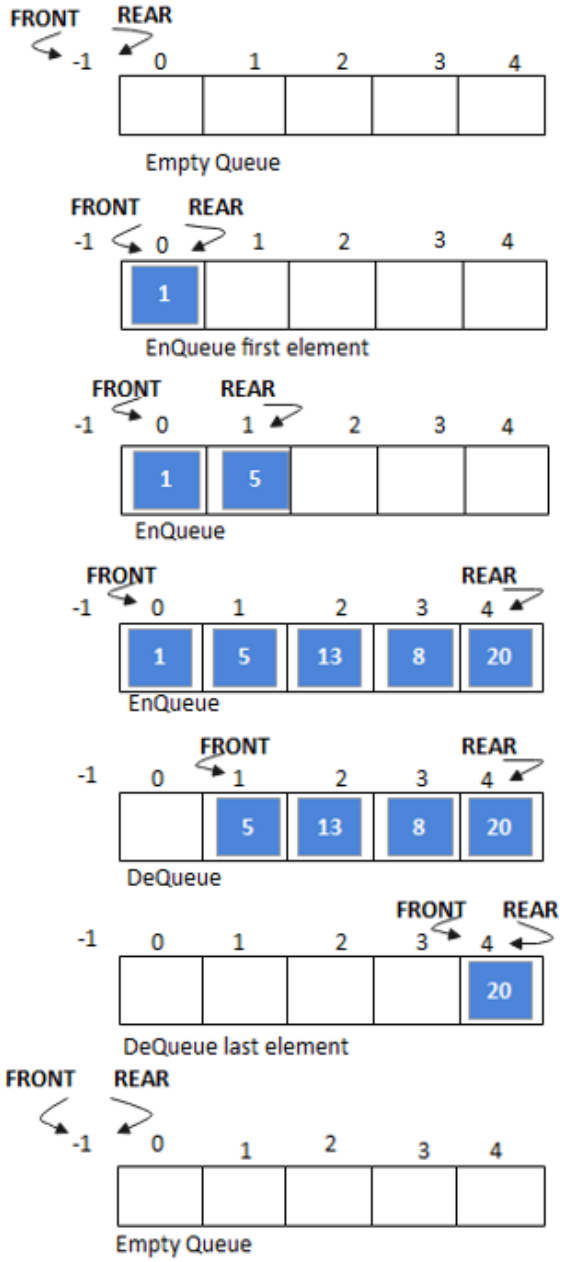


Figure 1: Queue operations illustrated

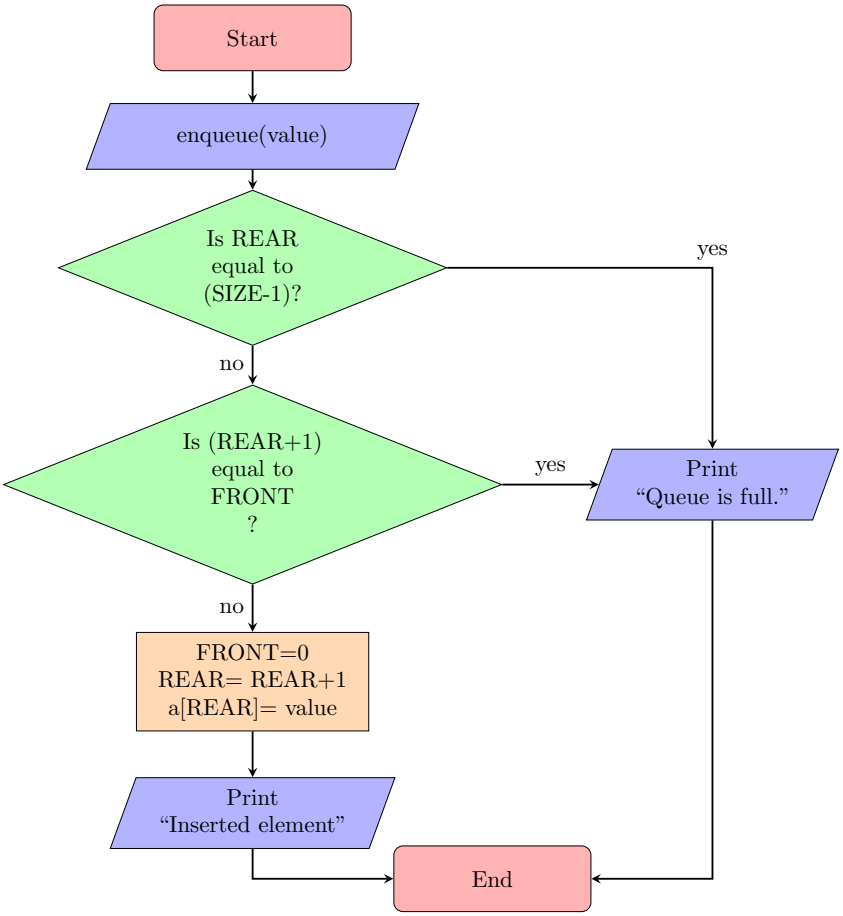
3 Algorithm

Algorithm 1: Enqueue Operation of Queue

Input: Element

```
/* Algorithm for enqueue operation                                    */
if REAR == SIZE-1 then
    Print "Queue is full"
    return
end
else
    if REAR + 1 == FRONT then
        Print "Queue is full"
        return
    end
    else
        FRONT = 0
        REAR = REAR + 1
        a[REAR] = value
        Print "Inserted element"
    end
end
```

3.1 Flowchart

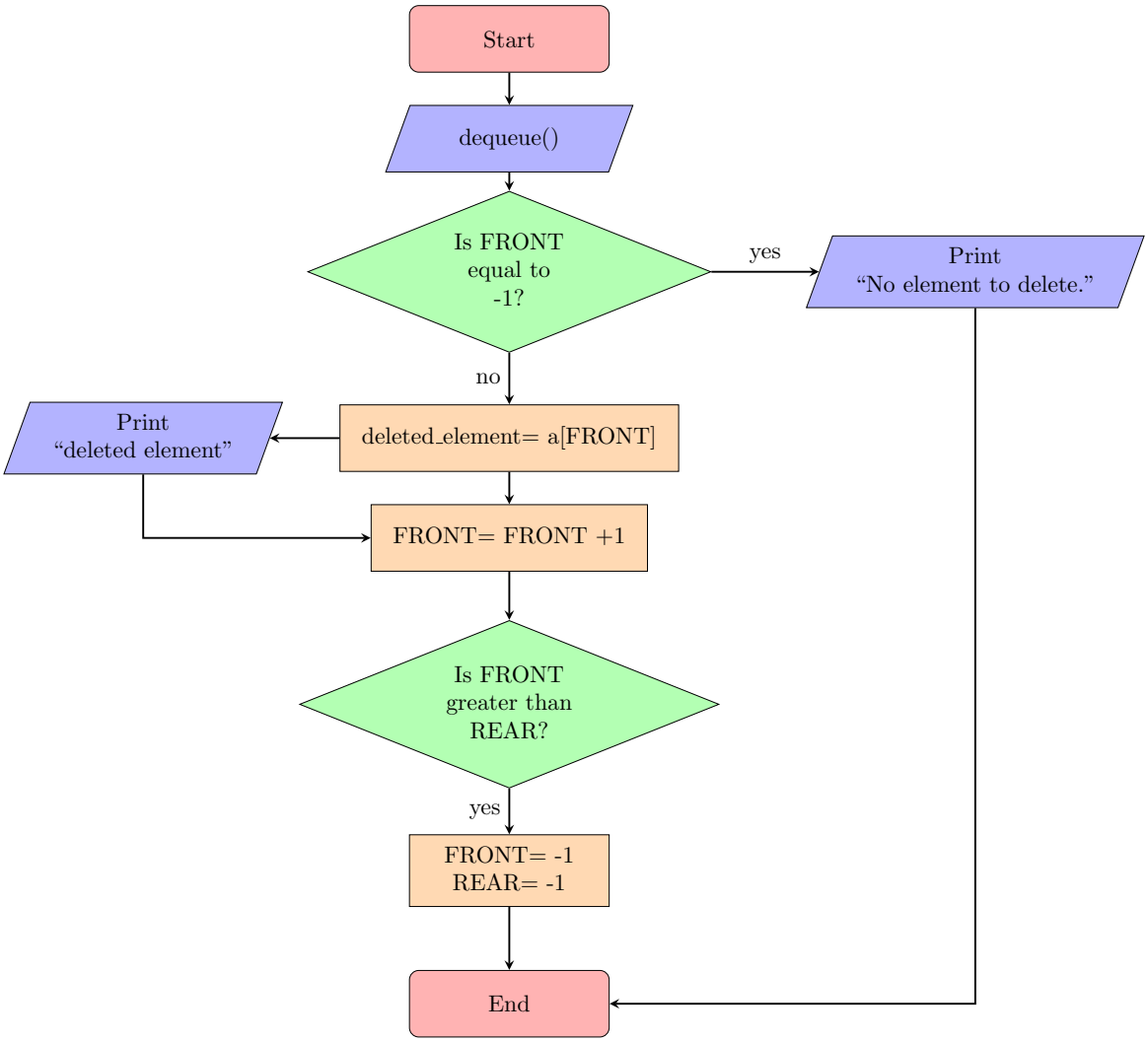


Algorithm 2: Dequeue Operation of Queue

Output: Element

```
/* Algorithm for dequeue operation                                    */
if FRONT == -1 then
    Print "No element to delete."
    return
end
else
    Set Del_element = a[FRONT]
    FRONT = FRONT+1
    return Del_Element
    if FRONT > REAR then
        FRONT = -1
        REAR = -1
    end
end
```

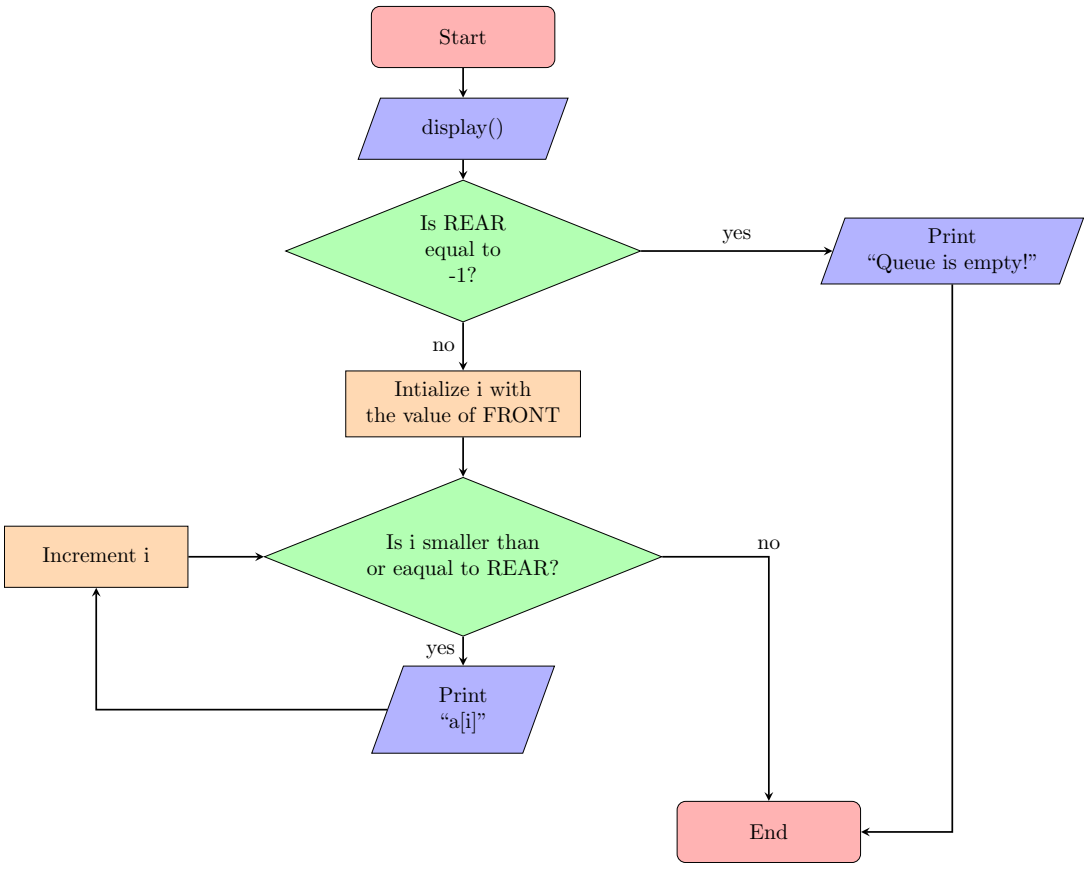
3.2 Flowchart



Algorithm 3: Display Operation of Queue

```
/* Algorithm for display operation                                    */
if REAR = -1 then
    print "Queue is empty!"
    return
end
else
    Print "Elements in the queue"
    for i = FRONT to REAR do
        print "a[i]"
    end
end
```

3.3 Flowchart



4 Implementation in C

```c
#include <stdio.h>
#include<stdlib.h>
#define size 5
int a[size],i,j,front=-1,rear=-1;

void enqueue(int value){
    if(rear==size-1)
        printf( "Queue is full.\n");
```

```
    else
    {
        if (rear+1==front)
            printf( "Queue is full.\n");
        else{
            front=0;
            rear++;
            a[rear]=value;
            printf("Inserted element: %d\n",a[rear]);
        }
    }
}

void dequeue(){
    if(front==-1){
        printf("No element to delete.\n");
    }
    else{
        printf("Deleted element: %d\n",a[front]);
        front=front+1;
        if(front>rear)
            front=rear=-1;
    }
}

void display(){
    if(rear==-1){
        printf("Queue is empty!\n");
    }
    else{
        printf("Elements in the queue: ");
        for(int i=front;i<=rear;i++){
            printf("%d ",a[i]);
        }
    }
}
int main(int argc, char const *argv[]) {
    int ch,val;
    do{
    printf("\nMenu");
    printf("\n1. ENQUEUE");
    printf("\n2. DEQUEUE ");
    printf("\n3. DISPLAY QUEUE");
    printf("\n4. Exit");
    printf("\nEnter your choice 1 to 4=");
    scanf("%d",&ch);
    switch (ch)
    {
        case 1:
                printf("Enter the value to be inserted=");
                scanf("%d",&val);
                enqueue(val);
                break;
        case 2:
                dequeue();
                break;

        case 3:
```

```
                display();
                break;
        case 4:
                exit(0);
                break;

        default:
                printf("Invalid choice!");
                break;
    }

    }while (ch<=3);

    return 0;
}
```

5 Input/Output (Compilation, Debugging & Testing)

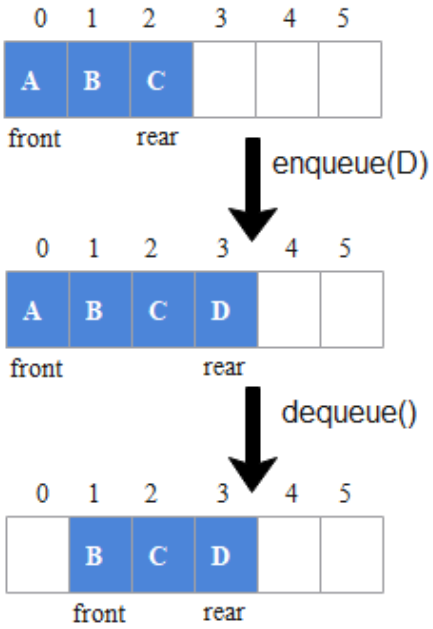
Input	Output
Choice: 3(initially)	Queue is empty.
Choice: 2(initially)	No element to delete.
Choice: 1 Enter the value to be inserted: 1	Inserted element: 1
Choice: 1 Enter the value to be inserted: 5	Inserted element: 5
Choice: 1 Enter the value to be inserted: 13	Inserted element: 13
Choice: 3	Elements in the queue: 1 5 13
Choice: 1 Enter the value to be inserted: 8	Inserted element: 8
Choice: 1 Enter the value to be inserted: 20	Inserted element: 20
Choice: 3	Elements in the queue: 1 5 13 8 20
Choice: 1 Enter the value to be inserted: 35	Queue is full
Choice: 3	Elements in the queue: 1 5 13 8 20
Choice: 2	Deleted element: 1
Choice: 3	Elements in the queue: 5 13 8 20
Choice: 2	Deleted element: 5
Choice: 2	Deleted element: 13
Choice: 3	Elements in the queue: 8 20
Choice: 2	Deleted element: 8
Choice: 3	Elements in the queue: 20
Choice: 2	Deleted element: 20
Choice: 3	Queue is empty
Choice: 2	No element to delete
Choice: 4	Press any key to continue.
Choice: 5	Invalid choice. Press any key to continue.

6 Discussion & Conclusion

Based on the focused objective(s) to understand about the queue operations, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

7 Lab Task (Please implement yourself and show the output to the instructor)

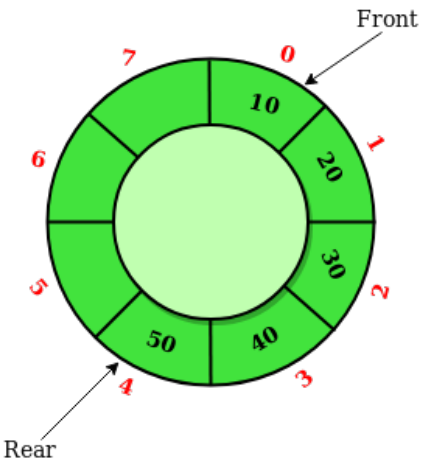
1. Implement the QUEUE shown in the following figure and perform the operations.



2. Apply enqueue() & dequeue() method.
3. Call a function that will generate the size of the queue.
4. Write appropriate code for controlling OVERFLOW and UNDERFLOW.

8 Lab Exercise (Submit as a report)

- Implement the following circular queue using arrays



9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.