SCS 1201

Data Structures and Algorithms

Answers - Tutorial 03

1.

Push Operation

Push(s, x)

operation's step are described below:
Enqueue x to q2
One by one dequeue everything from q1 and enqueue to q2.
Swap the names of q1 and q2

pop(s)

operation's function are described below
Dequeue an item from q1 and return it.

Pop operation

Push(s, x)

Enqueue x to q1 (assuming size of q1 is unlimited).

Pop(s)

One by one dequeue everything except the last element from q1 and enqueue to q2.
Dequeue the last item of q1, the dequeued item is result, store it.
Swap the names of q1 and q2
Return the item stored in step 2

Or

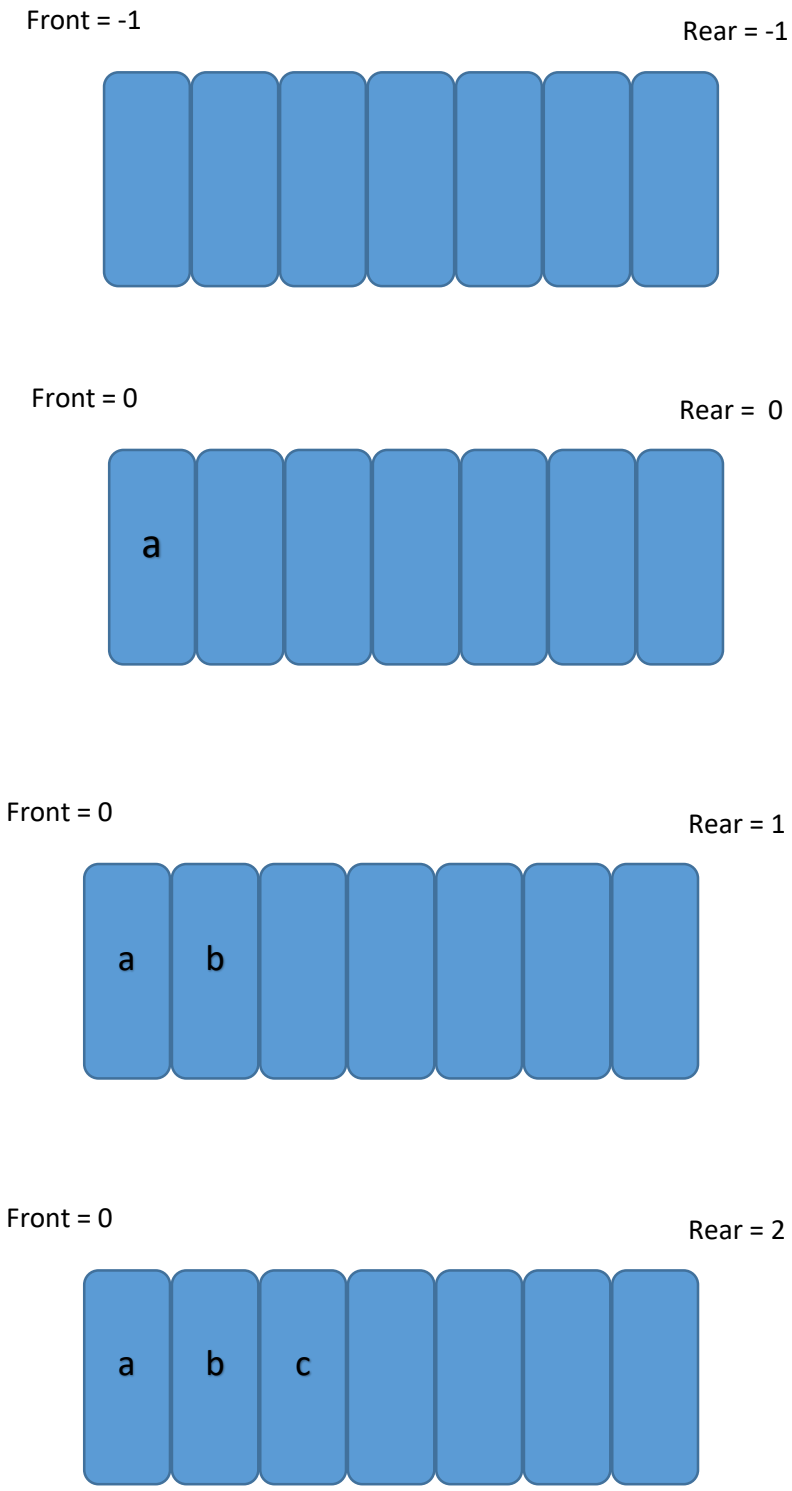
```
Begin
        set q1[max] ,q2[max]
        set front = -1 and rear = -1
        if user want to push an element
                while(rear<= max – 1)
                        if front =-1 and rear = -1
                                set front = rear = 0
                        else
                                set rear = rear + 1
                        end if
                        set q1[rear]  = num
                End while loop
                return q1
        else if user want to pop an element
                if front = -1 or front >rear
                        Display “Underflow”
                else
                        while (max > 1)
                                while (front <max -1)
                                        dequeue elements from q1 and
                                        enqueue to  q2
                                end while loop
                                swap the names of q1 and q2
                                max = max – 1
                        end while loop
End
```

- C code is attached in this folder(SCS1201_T3_Q1)

C:\Users\Gimhani\Desktop\Q1.exe

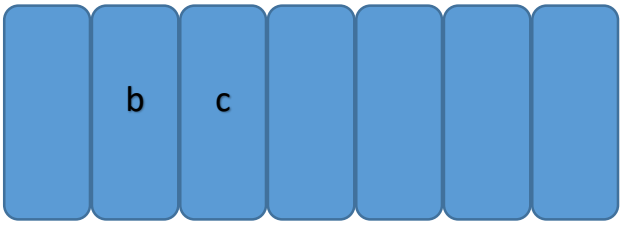
```
Choose one from below choices
  1. Push
  2. Pop
  3. Peek
  4. Exit
Choice : 1
Enter Value : 6


Choose one from below choices
  1. Push
  2. Pop
  3. Peek
  4. Exit
Choice : 1
Enter Value : 8


Choose one from below choices
  1. Push
  2. Pop
  3. Peek
  4. Exit
Choice : 2
Pop value = 8


Choose one from below choices
  1. Push
  2. Pop
  3. Peek
  4. Exit
Choice : 3
Peek Value = 6
```

2.


Front = -1
Rear = -1
Front = 0
Rear = 0
a
Front = 0
Rear = 1
a
b
Front = 0
Rear = 2
a
b
c

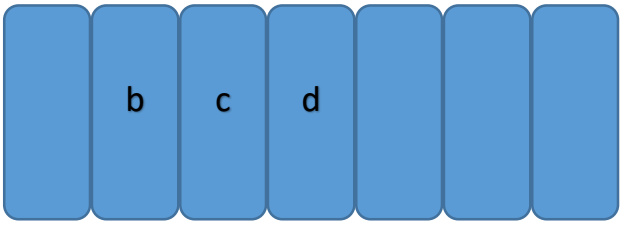
Front = 1

Rear = 2



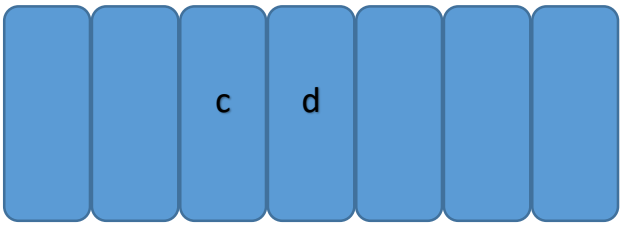
Front = 1

Rear = 3



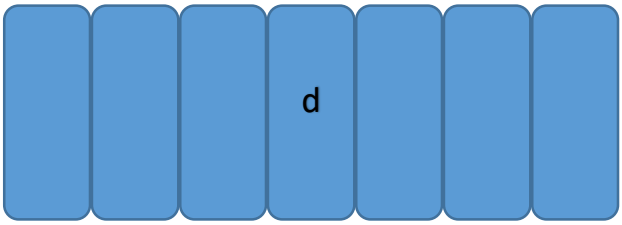
Front = 2

Rear = 3



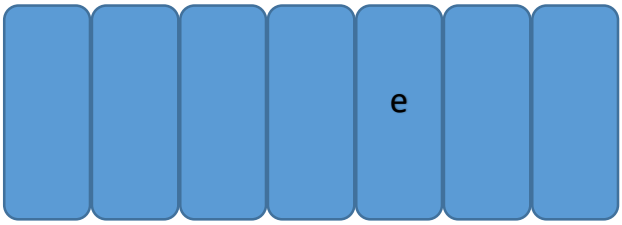
Front = 3

Rear = 3



Front = 4

Rear = 3



3. Begin

```
    declare a empty stack (stack1[max])
    declare a empty queue (q1)
    set top = 0
    while(top  <= max -1)
            stack1[top] = num
            top++
    end while loop
    set front = rear = -1
    while(top > -1)
            if (front = rear = -1)
                    front = rear = 0
                    q1[front] = stack[top]
            else
                    rear = rear +1
                    q1[rear] = stack [top]
            end if
            top = top – 1
     end while loop
    while (front < rear)
            print(q1[front])
            front = front – 1
    end while loop
End
```

- C code is attached in this folder(SCS1201_T3_Q3.cpp)

- outputs

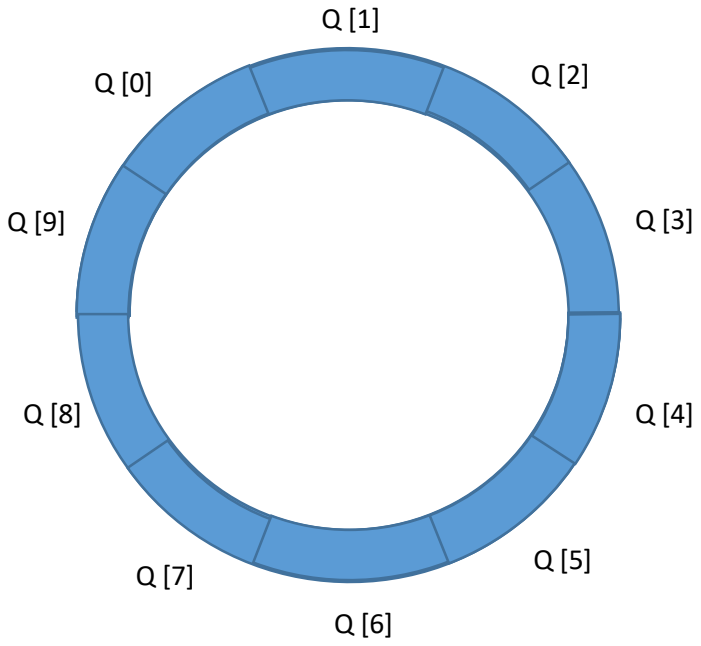
C:\Users\Gimhani\Desktop\SCS1201_T3_Q3.exe

```
how many number do you like to enter:10
Enter : 10
Enter : 20
Enter : 30
Enter : 40
Enter : 50
Enter : 60
Enter : 70
Enter : 80
Enter : 90
Enter : 100
100     90      80      70      60      50      40      30      20      10
--------------------------------
Process exited after 20.02 seconds with return value 0
Press any key to continue . . .
```

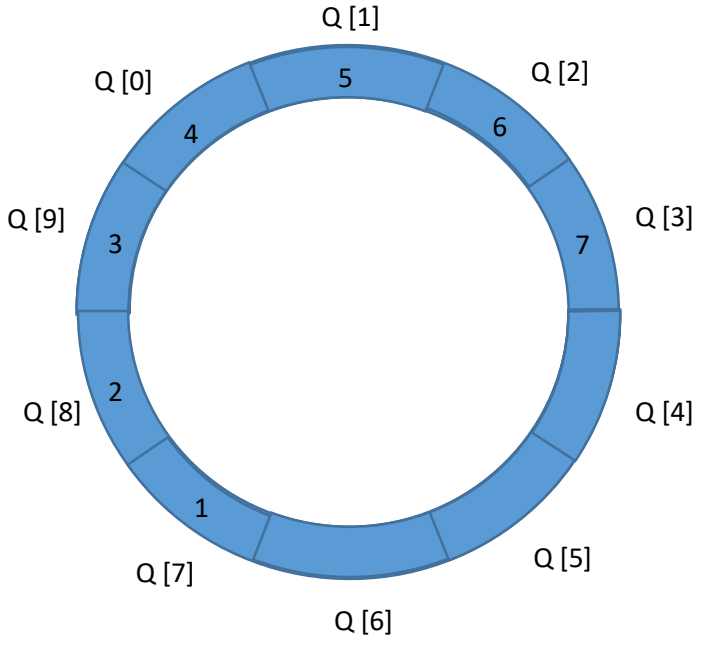
C:\Users\Gimhani\Desktop\SCS1201_T3_Q3.exe

```
how many number do you like to enter:5
Enter : 1
Enter : 2
Enter : 3
Enter : 4
Enter : 5
5       4       3       2       1
--------------------------------
Process exited after 6.126 seconds with return value 0
Press any key to continue . . .
```

4.



After the code segment is executed



5.

a) Output same set as same order.

b) 34, 12,18,23,55,11,9

6. Begin

```
        declare a empty queue as q1
        declare a integer variable y
        Get user input and set it as variable x
        n =0
        while(n<=x)
                y = n
                set front = rear = -1
                while(y >0)
                        if (front = rear = -1)
                                front = rear = 0
                                q1[front] = y%2
                        else
                                rear = rear +1
                                q1[rear] = y%2
                        end if
                        y = y/2
                end while loop
                while (front < rear)
                        print(q1[front])
                        front = front – 1
                end while loop
        end while loop
End
```

C code is attached in this folder (SCS1201_T3_Q6.cpp)

```
C:\Users\Gimhani\Desktop\SCS1201_T3_Q6.exe
Enter N : 10

1
10
11
100
101
110
111
1000
1001
1010

--------------------------------
Process exited after 2.492 seconds with return value 0
Press any key to continue . . .
```

7. Step 1 : If rear = max -1
    Print “Overflow”
    Go to step 4
  End if
  Step 2 : If front = -1 and rear = -1
    Set front = rear = 0
  Else
    Set rear = rear + 1
  End if
  Step 3 : Set Queue[rear] = num
  Step 4 : Exit

8. Both front and rear are increase but never decrease.
   As items are removed from the queue, the storage space at the beginning of the array is discarded and never used again.

9. Step 1 : If front = -1
    Print “Circular Queue Underflow”
    Go to step 4
  End if
  Step 2 : Return (CQ[front])
  Step 3 : If front = rear
    front = rear = -1
  End if
  Step 4 : If front = size – 1
    front = 0
  Else
    front = front + 1
  End if
  Step 5 : Exit

10. **EnQueue Costly**

EnQueue (q, x)

1) While stack1 is not empty, push everything from stack1 to stack2.
2) Push x to stack1 (assuming size of stacks is unlimited).
3) Push everything back to stack1.

Here time complexity will be O(n)

DeQueue (q)

1) If stack1 is empty then error

2) Pop an item from stack1 and return it

Here time complexity will be O(1)

DeQueue Costly

EnQueue (q, x)

1) Push x to stack1 (assuming size of stacks is unlimited).

Here time complexity will be O(1)

DeQueue (q)
1) If both stacks are empty then error.

2) If stack2 is empty

While stack1 is not empty, push everything from stack1 to stack2.

3) Pop the element from stack2 and return it.

Here time complexity will be O(n)