**Course: Data Structures and Algorithms Assignment 3**

**Queue and Binary Tree** *Dr. Belal Al-Fuhaidi*

1- Consider the following statements:

```
Array Queue<int> queue = new Array Queue();
int x, y;
```

Show what is output by the following segment of code:

```
x = 4;
y = 5;
queue.enqueue(x);
queue. enqueue(y);
x = queue.front( );
queue.dequeue( );
queue. enqueue(x + 5);
queue. enqueue(16);
queue. enqueue(x);
queue. enqueue(y - 3);
system.out.println( "Queue Elements: ");
while (!queue.isEmptyQueue())
{
system.out.println(queue.front() );
queue.dequeue();
}
```

5
9
16
4
2

2- What is the output of the following program segment?

```
linkedQueue<int> queue = new linkedQueue();
queue.enqueue(10);
queue.enqueue(20);
cout << queue.front() << endl;
queue.dequeue();
queue.enqueue(2 * queue.back());
queue.enqueue(queue.front());
queue. enqueue(5);
queue. enqueue(queue.back() - 2);
linkedQueue<int> tempQueue = new linkedQueue() ;
tempQueue = queue;
while (!tempQueue.isEmptyQueue())
{
   system.out.println( tempQueue.front() );
   tempQueue.dequeue();
}
system.out.println( queue.front() );
system.out.println(queue.back() );
```

10
20
40
20
5
3
null
null

3- Consider the following statements:

```
ArrayStack<int> stack = new ArrayStack();
ArrayQueue<int> queue = new ArrayQueue();
int x;
```

Suppose the input is:

**Course: Data Structures and Algorithms Assignment 3**

**Queue and Binary Tree** *Dr. Belal Al-Fuhaidi*

14 8 14 22 64 35 19 32 7 11 13 30 -999

Show what is written by the following segment of code:

```
stack.push(0);
queue.enqueue(0);
system.out.println( x);
while (x != -999)
{
    switch (x % 4)
    {
    case 0:
        stack.push(x);
        break;
    case 1:
    if (!stack.isEmptyStack())
    {
    system.out.println( "Stack Element = " );
    system.out.println( stack.top());
    stack.pop();
    }
    else
      system.out.println(  "Sorry, the stack is empty." );
    break;
    case 2:
    queue.enqueue(x);
    break;
    case 3:
    if (!queue.isEmptyQueue())
    {
    system.out.println( "Queue Element = " );
    system.out.println( queue.front());
    queue.dequeue();
    }
    else
    system.out.println(  "Sorry, the queue is empty." );
    break;
    } //end switch
system.out.println( x);
  } //end while
    system.out.println(  "Stack Elements: ");
    while (!stack.isEmptyStack())
    {
    system.out.println( stack.top() );
    stack.pop();
    }
    system.out.println(  "Queue Elements: ");
    while (!queue.isEmptyQueue())
    {
```

14,8,14,22,64,35,19,32,7,11,13,30,-999

14,8,14,22,64,35,19,32,7,11,13,30

1,3

8,6,14,32

14,14,22,30

1,35,19,7,11

14,8,14,22,64,35,7,11,13,30

22
11
stack Element
32
13
30

stack Element
64
8
0
queueElement
30

output
14
14
8
14
22
64
queueElement
0

35
queue Element
14
19
32
queue Element
14
7
queueElement

```
system.out.println( queue.front() );
queue.dequeue();
}
```

4- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 50 and the value of queueRear is 99.
   a- What are the values of queueFront and queueRear after adding an element to queue? queue front = 50 queue Rear = 0
   b- What are the values of queueFront and queueRear after removing an element from queue? queue front = 50 queue Root = 0
   after queue front = 51 " = 0

5- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 99 and the value of queueRear is 25.
   a- What are the values of queueFront and queueRear after adding an element to queue? queue front = 99 queue Root = 25, after queu front = 99
   b- What are the values of queueFront and queueRear after removing an element from queue? " " = 99 " " = 26, " " = 0 " root = 26

6- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 25 and the value of queueRear is 75. queue root = 26
   a- What are the values of queueFront and queueRear after adding an element to queue? queue front = 25 queue root = 75, after queue = 25
   b- What are the values of queueFront and queueRear after removing an element from queue? " " = 25 " " = 75, " " = 26 " root = 76 " 76

7- Suppose that queue is a queueType object and the size of the array implementing queue is 100. Also, suppose that the value of queueFront is 99 and the value of queueRear is 99.
   a- What are the values of queueFront and queueRear after adding an element to queue? queue front = 99 queue root = 99, after queue = 99
   b- What are the values of queueFront and queueRear after removing an element from queue? " " = 99 " " = 0, " " = 0 " root = 0 " " = 0

8- Write a function, **reverseQueue**, that takes as a parameter a queue object and uses a stack object to reverse the elements of the queue. ⟹

9- Suppose an initially empty queue *Q* has performed a total of 32 enqueue operations, 10 first operations, and 15 dequeue operations, 5 of which returned null to indicate an empty queue. What is the current size of *Q*? current size of Q = 32-(15-5) = 32-10 = 22

10- What values are returned during the following sequence of deque *(double ended queue)* ADT operations, on an initially empty deque? addFirst(3), addLast(8), addLast(9), addFirst(1), last( ), isEmpty( ), addFirst(2), removeLast( ), addLast(7), first( ), last( ), addLast(4), size( ), removeFirst( ), removeFirst( ).

⟹

**Good Luck**

```
public class linkedQueue<E> implemts intaf Queue<E>
// LinkedQueue<E> reverseQueue(LinkedQueue<E> queue;
ArrayStack<E> s=new ArrayStack<>(queue.size));
While(!s.isEmpty())
queue.enqueue
queue.enqueue(s.pop());
returnqueue;}}

Public class Q8{
Public static Void main(sting[] array){
LinkedQueue<integer> q=new linkedQueue<>();
q.enqueue(10); q.enqueue(20);
// (30); // (40);
q.Print(q); System.out.print();
q.nereveQueue(q);
```

Output:

10 20 30 40

40 30 20 10

| method | rethun | ID |
|---|---|---|
| addfirst(3) | - | (3) |
| // Last(8) | - | (3,8) |
| // //(9) | - | (3,8,9) |
| // first(1) | - | (1,3,8,9) |
| Last() | 9 | (1,3,8,9) |
| isEmply() | False | (1,3,8,9) |
| add first(2) | - | (2,1,3,8,9) |
| removeLast() | - | (2,1,3,8,9) |
| addlast(7) | - | (2,1,3,8,7) |
| fiyst() | 2 | (2,1,3,8,7) |
| Last() | 7 | (2,1,3,8,7) |
| add Last(4) | - | (2,1,3,8,7,4) |
| size() | 6 | (2,1,3,8,7,4) |
| removefirst() | 2 | (1,3,8,7,4) |
| // // | 1 | (3,8,7,4) |