“Queue using Array”.

1. Which of the following properties is associated with a queue?
a) First In Last Out
b) First In First Out
c) Last In First Out
d) Last In Last Out
View Answer
Answer: b
Explanation: Queue follows First In First Out structure.

2. In a circular queue, how do you increment the rear end of the queue?
a) rear++
b) (rear+1) % CAPACITY
c) (rear % CAPACITY)+1
d) rear–
View Answer
Answer: b
Explanation: Ensures rear takes the values from 0 to (CAPACITY-1).

3. What is the term for inserting into a full queue known as?
a) overflow
b) underflow
c) null pointer exception
d) program won’t be compiled
View Answer
Answer: a
Explanation: Just as stack, inserting into a full queue is termed overflow.

4. What is the time complexity of enqueue operation?
a) O(logn)
b) O(nlogn)
c) O(n)
d) O(1)
View Answer
Answer: d
Explanation: Enqueue operation is at the rear end, it takes O(1) time to insert a new item into the queue.

5. What does the following Java code do?

```
public Object function()
```

```
{
        if(isEmpty())
        return -999;
        else
        {
                Object high;
                high = q[front];
                return high;
        }
}
```

a) Dequeue
b) Enqueue
c) Return the front element
d) Return the last element
View Answer
Answer: c
Explanation: q[front] gives the element at the front of the queue, since we are not moving the ‘front’ to the next element,
it is not a dequeue operation.

6. What is the need for a circular queue?
a) effective usage of memory
b) easier computations
c) to delete elements based on priority
d) implement LIFO principle in queues
View Answer
Answer: a
Explanation: In a linear queue, dequeue operation causes the starting elements of the array to be empty, and there is no way you can use that space, while in a circular queue, you can effectively use that space. Priority queue is used to delete the elements based on their priority. Higher priority elements will be deleted first whereas lower priority elements will be deleted next. Queue data structure always follows FIFO principle.

7. Which of the following represents a dequeue operation? (count is the number of elements in the queue)
a)

```
public Object dequeue()
{
        if(count == 0)
        {
                System.out.println("Queue underflow");
                return 0;
        }
        else
        {
                Object ele = q[front];
                q[front] = null;
                front = (front+1)%CAPACITY;
                count--;
                return ele;
        }
}
```

b)

```
public Object dequeue()
{
        if(count == 0)
        {
                System.out.println("Queue underflow");
                return 0;
        }
        else
        {
                Object ele = q[front];
                front = (front+1)%CAPACITY;
                q[front] = null;
                count--;
                return ele;
        }
}
```

c)

```
public Object dequeue()
{
        if(count == 0)
        {
                System.out.println("Queue underflow");
                return 0;
        }
        else
        {
                front = (front+1)%CAPACITY;
                Object ele = q[front];
                q[front] = null;
                count--;
                return ele;
        }
}
```

d)

```
public Object dequeue()
{
        if(count == 0)
        {
                System.out.println("Queue underflow");
                return 0;
        }
        else
        {
                Object ele = q[front];
                q[front] = null;
                front = (front+1)%CAPACITY;
                return ele;
                count--;
        }
}
```

View Answer

Answer: a

Explanation: Dequeue removes the first element from the queue, 'front' points to the front end of the queue and returns the first element.

8. Which of the following best describes the growth of a linear queue at runtime? (Q is the original queue, size() returns the number of elements in the queue)

a)

```
private void expand()
{
        int length = size();
        int[] newQ = new int[length<<1];
        for(int i=front; i<=rear; i++)
```

```
         {
                  newQ[i-front] = Q[i%CAPACITY];
         }
         Q = newQ;
         front = 0;
         rear = size()-1;
}
```

b)

```
private void expand()
{
         int length = size();
         int[] newQ = new int[length<<1];
         for(int i=front; i<=rear; i++)
         {
                  newQ[i-front] = Q[i%CAPACITY];
         }
         Q = newQ;
}
```

c)

```
private void expand()
{
         int length = size();
         int[] newQ = new int[length<<1];
         for(int i=front; i<=rear; i++)
         {
                  newQ[i-front] = Q[i];
         }
         Q = newQ;
         front = 0;
         rear = size()-1;
}
```

d)

```
private void expand()
{
         int length = size();
         int[] newQ = new int[length*2];
         for(int i=front; i<=rear; i++)
         {
                  newQ[i-front] = Q[i%CAPACITY];
         }
         Q = newQ;
}
```

View Answer

Answer: a

Explanation: A common technique to expand the size of array at run time is simply to double the size. Create a new array of double the previous size and copy all the elements, after copying do not forget to assign front = 0 and rear = size()-1, as these are necessary to maintain the decorum of the queue operations.

9. What is the space complexity of a linear queue having n elements?
a) O(n)
b) O(nlogn)
c) O(logn)
d) O(1)

View Answer

Answer: a

Explanation: Because there are n elements.

10. What is the output of the following Java code?

```
public class CircularQueue
{
         protected static final int CAPACITY = 100;
         protected int size,front,rear;
         protected Object q[];
         int count = 0;

         public CircularQueue()
         {
                  this(CAPACITY);
         }
         public CircularQueue (int n)
         {
                  size = n;
                  front = 0;
                  rear = 0;
                  q = new Object[size];
```

```
        }


        public void enqueue(Object item)
        {
                if(count == size)
                {
                        System.out.println("Queue overflow");
                                return;
                }
                else
                {
                        q[rear] = item;
                        rear = (rear+1)%size;
                        count++;
                }
        }
        public Object dequeue()
        {
                if(count == 0)
                {
                        System.out.println("Queue underflow");
                        return 0;
                }
                else
                {
                        Object ele = q[front];
                        q[front] = null;
                        front = (front+1)%size;
                        count--;
                        return ele;
                }
        }
        public Object frontElement()
        {
                if(count == 0)
                return -999;
                else
                {
                        Object high;
                        high = q[front];
                        return high;
                }
```

```
         }
         public Object rearElement()
         {
                  if(count == 0)
                  return -999;
                  else
                  {
                           Object low;
                           rear = (rear-1)%size;
                           low = q[rear];
                           rear = (rear+1)%size;
                           return low;
                  }
         }
}
public class CircularQueueDemo
{
         public static void main(String args[])
         {
                  Object var;
                  CircularQueue myQ = new CircularQueue();
                  myQ.enqueue(10);
                  myQ.enqueue(3);
                  var = myQ.rearElement();
                  myQ.dequeue();
                  myQ.enqueue(6);
                  var = mQ.frontElement();
                  System.out.println(var+" "+var);
         }
}
```

a) 3 3
b) 3 6
c) 6 6
d) 10 6

View Answer

Answer: a

Explanation: First enqueue 10 and 3 into the queue, followed by a dequeue(removes 10), followed by an enqueue(6), At this point, 3 is at the front end of the queue and 6 at the rear end, hence a call to frontElement() will return 3 which is displayed twice.

“Queue using Linked List”.

1. In linked list implementation of queue, if only front pointer is maintained, which of the following operation take worst case linear time?
a) Insertion
b) Deletion
c) To empty a queue
d) Both Insertion and To empty a queue
View Answer
Answer: d
Explanation: Since front pointer is used for deletion, so worst time for the other two cases.

2. In linked list implementation of a queue, where does a new element be inserted?
a) At the head of link list
b) At the centre position in the link list
c) At the tail of the link list
d) At any position in the linked list
View Answer
Answer: c
Explanation: Since queue follows FIFO so new element inserted at last.

3. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a NONEMPTY queue?
a) Only front pointer
b) Only rear pointer
c) Both front and rear pointer
d) No pointer will be changed
View Answer
Answer: b
Explanation: Since queue follows FIFO so new element inserted at last.

4. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into EMPTY queue?
a) Only front pointer
b) Only rear pointer
c) Both front and rear pointer
d) No pointer will be changed
View Answer
Answer: c
Explanation: Since its the starting of queue, so both values are changed.

5. In case of insertion into a linked queue, a node borrowed from the __________ list is inserted in the queue.
a) AVAIL
b) FRONT
c) REAR
d) NULL
View Answer
Answer: a
Explanation: All the nodes are collected in AVAIL list.

6. In linked list implementation of a queue, from where is the item deleted?
a) At the head of link list
b) At the centre position in the link list
c) At the tail of the link list
d) Node before the tail
View Answer
Answer: a
Explanation: Since queue follows FIFO so new element deleted from first.

7. In linked list implementation of a queue, the important condition for a queue to be empty is?
a) FRONT is null
b) REAR is null
c) LINK is empty
d) FRONT==REAR-1
View Answer
Answer: a
Explanation: Because front represents the deleted nodes.

8. The essential condition which is checked before insertion in a linked queue is?
a) Underflow
b) Overflow
c) Front value
d) Rear value
View Answer
Answer: b
Explanation: To check whether there is space in the queue or not.

9. The essential condition which is checked before deletion in a linked queue is?
a) Underflow
b) Overflow
c) Front value

d) Rear value
View Answer
Answer: a
Explanation: To check whether there is element in the list or not.

10. Which of the following is true about linked list implementation of queue?
a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end
b) In push operation, if new nodes are inserted at the beginning, then in pop operation, nodes must be removed from the beginning
c) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from end
d) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from beginning
View Answer
Answer: a
Explanation: It can be done by both the methods.

"Queue using Stacks".

1. A Double-ended queue supports operations such as adding and removing items from both the sides of the queue. They support four operations like addFront(adding item to top of the queue), addRear(adding item to the bottom of the queue), removeFront(removing item from the top of the queue) and removeRear(removing item from the bottom of the queue). You are given only stacks to implement this data structure. You can implement only push and pop operations. What are the total number of stacks required for this operation?(you can reuse the stack)
a) 1
b) 2
c) 3
d) 4
View Answer
Answer: b
Explanation: The addFront and removeFront operations can be performed using one stack itself as push and pop are supported (adding and removing element from top of the stack) but to perform addRear and removeRear you need to pop each element from the current stack and push it into another stack, push or pop the element as per

the asked operation from this stack and in the end pop elements from this stack to the first stack.

2. You are asked to perform a queue operation using a stack. Assume the size of the stack is some value 'n' and there are 'm' number of variables in this stack. The time complexity of performing deQueue operation is (Using only stack operations like push and pop)(Tightly bound).
a) O(m)
b) O(n)
c) O(m*n)
d) Data is insufficient
View Answer
Answer: a
Explanation: To perform deQueue operation you need to pop each element from the first stack and push it into the second stack. In this case you need to pop 'm' times and need to perform push operations also 'm' times. Then you pop the first element from this second stack (constant time) and pass all the elements to the first stack (as done in the beginning)('m-1' times). Therfore the time complexity is O(m).

3. Consider you have an array of some random size. You need to perform dequeue operation. You can perform it using stack operation (push and pop) or using queue operations itself (enQueue and Dequeue). The output is guaranteed to be same. Find some differences?
a) They will have different time complexities
b) The memory used will not be different
c) There are chances that output might be different
d) No differences
View Answer
Answer: a
Explanation: To perform operations such as Dequeue using stack operation you need to empty all the elements from the current stack and push it into the next stack, resulting in a O(number of elements) complexity whereas the time complexity of dequeue operation itself is O(1). And there is a need of a extra stack. Therefore more memory is needed.

4. Consider you have a stack whose elements in it are as follows.
5 4 3 2 << top
Where the top element is 2.
You need to get the following stack
6 5 4 3 2 << top
The operations that needed to be performed are (You can perform only push and pop):
a) Push(pop()), push(6), push(pop())

b) Push(pop()), push(6)
c) Push(pop()), push(pop()), push(6)
d) Push(6)
View Answer
Answer: a
Explanation: By performing push(pop()) on all elements on the current stack to the next stack you get 2 3 4 5 << top.Push(6) and perform push(pop()) you'll get back 6 5 4 3 2 << top. You have actually performed enQueue operation using push and pop.

5. A double-ended queue supports operations like adding and removing items from both the sides of the queue. They support four operations like addFront(adding item to top of the queue), addRear(adding item to the bottom of the queue), removeFront(removing item from the top of the queue) and removeRear(removing item from the bottom of the queue). You are given only stacks to implement this data structure. You can implement only push and pop operations. What's the time complexity of performing addFront and addRear? (Assume 'm' to be the size of the stack and 'n' to be the number of elements)
a) O(m) and O(n)
b) O(1) and O(n)
c) O(n) and O(1)
d) O(n) and O(m)
View Answer
Answer: b
Explanation: addFront is just a normal push operation. Push operation is of O(1). Whereas addRear is of O(n) as it requires two push(pop()) operations of all elements of a stack.

6. Why is implementation of stack operations on queues not feasible for a large dataset (Asssume the number of elements in the stack to be n)?
a) Because of its time complexity O(n)
b) Because of its time complexity O(log(n))
c) Extra memory is not required
d) There are no problems
View Answer
Answer: a
Explanation: To perform Queue operations such as enQueue and deQueue there is a need of emptying all the elements of a current stack and pushing elements into the next stack and vice versa. Therfore it has a time complexity of O(n) and the need of extra stack as well, may not be feasible for a large dataset.

7. Consider yourself to be in a planet where the computational power of chips to be slow. You have an array of size 10.You want to perform enqueue some element into

this array. But you can perform only push and pop operations .Push and pop operation both take 1 sec respectively. The total time required to perform enQueue operation is?
a) 20
b) 40
c) 42
d) 43
View Answer
8. You have two jars, one jar which has 10 rings and the other has none. They are placed one above the other. You want to remove the last ring in the jar. And the second jar is weak and cannot be used to store rings for a long time.
a) Empty the first jar by removing it one by one from the first jar and placing it into the second jar
b) Empty the first jar by removing it one by one from the first jar and placing it into the second jar and empty the second jar by placing all the rings into the first jar one by one
c) There exists no possible way to do this
d) Break the jar and remove the last one
View Answer
Answer: b
Explanation: This is similar to performing dequeue operation using push and pop only. Elements in the first jar are taken out and placed in the second jar. After removing the last element from the first jar, remove all the elements in the second jar and place them in the first jar.

9. Given only a single array of size 10 and no other memory is available. Which of the following operation is not feasible to implement (Given only push and pop operation)?
a) Push
b) Pop
c) Enqueue
d) Returntop
View Answer
Answer: c
Explanation: To perform Enqueue using just push and pop operations, there is a need of another array of same size. But as there is no extra available memeory, the given operation is not feasible.

10. Given an array of size n, let's assume an element is 'touched' if and only if some operation is performed on it(for example, for performing a pop operation the top element is 'touched'). Now you need to perform Dequeue operation. Each element in the array is touched atleast?
a) Once
b) Twice
c) Thrice

d) Four times
View Answer
Answer: d
Explanation: First each element from the first stack is popped, then pushed into the second stack, dequeue operation is done on the top of the stack and later the each element of second stack is popped then pushed into the first stack. Therfore each element is touched four times.