SANFOUNDRY
Global Education & Learning

Data Structure Questions and Answers – Queue using Array

« Prev Next »

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “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 piece of 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 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 piece of 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.

Sanfoundry Global Education & Learning Series – Data Structure.

To practice all areas of Data Structure, here is complete set of 1000+ Multiple Choice Questions and Answers.

« Prev - Data Structure Questions and Answers – Stack using Linked List

» Next - Data Structure Questions and Answers – Queue using Linked List

Recommended Posts:

1. C Programming Examples
2. Data Structures & Algorithms II – Questions and Answers
3. C Programming Examples without using Recursion
4. Python Programming Examples on Trees
5. Java Programming Examples on Multithreading
6. C Programming Examples on Trees

7. Data Science Questions and Answers
8. Java Programming Examples on Arrays
9. C++ Programming Examples on STL
10. Java Programming Examples on Collections
11. C# Programming Examples on Data Structures
12. Python Programming Examples on Linked Lists
13. C Programming Examples on Linked List
14. C Programming Examples on Arrays
15. C Programming Examples on Data-Structures
16. C++ Programming Examples on Data-Structures
17. Java Programming Examples on Data-Structures
18. Python Programming Examples on Stacks & Queues
19. C Programming Examples on Stacks & Queues
20. Data Structure Questions and Answers

Manish Bhojasia, a technology veteran with 20+ years @ Cisco & Wipro, is Founder and CTO at Sanfoundry. He is Linux Kernel Developer & SAN Architect and is passionate about competency developments in these areas. He lives in Bangalore and delivers focused training sessions to IT professionals in Linux Kernel, Linux Debugging, Linux Device Drivers, Linux Networking, Linux Storage, Advanced C Programming, SAN Storage Technologies, SCSI Internals & Storage Protocols such as iSCSI & Fiber Channel. Stay connected with him @ LinkedIn