This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack using Array”.

1. Which of the following real world scenarios would you associate with a stack data structure?

a) piling up of chairs one above the other

b) people standing in a line to be serviced at a counter

c) offer services based on the priority of the customer

d) tatkal Ticket Booking in IRCTC

Answer: a

Explanation: Stack follows Last In First Out (LIFO) policy. Piling up of chairs one above the other is based on LIFO, people standing in a line is a queue and if the service is based on priority, then it can be associated with a priority queue. Tatkal Ticket Booking Follows First in First Out Policy. People who click the book now first will enter the booking page first.

2. What does the following function check for? (all

necessary headers to be included and function is called from main)

```
#define MAX 10

typedef struct stack
{
	int top;
	int item[MAX];
}stack;

int function(stack *s)
{
	if(s->top == -1)
		return 1;
	else return 0;
}
```

a) full stack

b) invalid index

c) empty stack

d) infinite stack

3. What does ‘stack underflow’ refer to?

a) accessing item from an undefined stack

b) adding items to a full stack

c) removing items from an empty stack

d) index out of bounds exception

4. What is the output of the following program?

```
public class Stack
{
```

```
protected static final int CAPACITY = 100;
protected int size,top = -1;
protected Object stk[];

public Stack()
{
	stk = new Object[CAPACITY];
}

public void push(Object item)
{
	if(size_of_stack==size)
	{
		System.out.println("Stack overflow");
			return;
	}
	else
```

```
        {
            top++;
            stk[top]=item;
        }
}
public Object pop()
{
        if(top<0)
        {
            return -999;
        }
        else
        {
            Object ele=stk[top];
            top--;
            size_of_stack--;
            return ele;
```

```
            }
        }
}

public class StackDemo
{
        public static void main(String args[])
        {
                Stack myStack = new Stack();
                myStack.push(10);
                Object element1 = myStack.pop();
                Object element2 = myStack.pop();
                System.out.println(element2);
        }
}
```

a) stack is full

b) 20

c) 0

d) -999

View Answer

Answer: d

Explanation: The first call to pop() returns 10, whereas the second call to pop() would result in stack underflow and the program returns -999.

5. What is the time complexity of pop() operation when the stack is implemented using an array?

a) O(1)

b) O(n)

c) O(logn)

d) O(nlogn)

6. Which of the following array position will be occupied by a new element being pushed for a stack of size N elements(capacity of stack > N)?

a) S[N-1]

b) S[N]

c) S[1]

d) S[0]

7. What happens when you pop from an empty stack while implementing using the Stack ADT in Java?

a) Undefined error

b) Compiler displays a warning

c) EmptyStackException is thrown

d) NoStackException is thrown

View Answer

8. What is the functionality of the following piece of Java code?

Assume: ‘a’ is a non empty array of integers, the Stack class creates an array of specified size and provides a top

pointer indicating TOS(top of stack), push and pop have normal meaning.

```
public void some_function(int[] a)
{
    Stack S=new Stack(a.length);
    int[] b=new int[a.length];
    for(int i=0;i<a.length;i++)
    {
        S.push(a[i]);
    }
    for(int i=0;i<a.length;i++)
    {
        b[i]=(int)(S.pop());
    }
    System.out.println("output :");
    for(int i=0;i<b.length;i++)
```

```
    {
        System.out.println(b[i]);
    }
}
```

a) print alternate elements of array

b) duplicate the given array

c) parentheses matching

d) reverse the array

9. Array implementation of Stack is not dynamic, which of the following statements supports this argument?

a) space allocation for array is fixed and cannot be changed during run-time

b) user unable to give the input for stack operations

c) a runtime exception halts execution

d) improper program compilation

View Answer

10. Which of the following array element will return the top-of-the-stack-element for a stack of size N elements(capacity of stack > N)?

a) S[N-1]

b) S[N]

c) S[N-2]

d) S[N+1]

Answer: a

Explanation: Array indexing start from 0, hence N-1 is the last index.

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

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–

What is the term for inserting into a full queue known .3 ?as

a) overflow

b) underflow

c) null pointer exception

d) program won’t be compiled

?What is the time complexity of enqueue operation .4

a) O(logn)

b) O(nlogn)

c) O(n)

d) O(1)

?What does the following Java code do .5

!Get Free Certificate of Merit in Data Structure I Now

()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

?What is the need for a circular queue .6

a) effective usage of memory

b) easier computations

c) to delete elements based on priority

d) implement LIFO principle in queues

View Answer

Which of the following represents a dequeue .7 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
    {
{
```

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

Which of the following best describes the growth of a .8 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

What is the space complexity of a linear queue having .9
?n elements

a) O(n)

b) O(nlogn)

c) O(logn)

d) O(1)

View Answer

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

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