AMCAT Stacks and Queues Questions

Question1

These operations can be performed on which type of structure? Push, Pop, Peek

A. Queue

B. Priority Queue

C. Stack

D. Both 1 and 2

Answer: Option **D**

Explanation: Only stack has these queue has enqueue and dequeue

Question 2

Which one of the following is an application of Queue Data Structure?

A. When a resource is shared among multiple consumers.

B. When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes

C. Load Balancing

D. All of the above

Answer: Option **D**

Question 3

How many stacks are needed to implement a queue. Consider the situation where no other data structure like arrays, linked list is available to you.

A. 1

B. 2

C. 3

D. 4

Answer: Option **B**

Question 4

A priority queue can efficiently implemented using which of the following data structures? Assume that the number of insert and peek (operation to see the current highest priority item) and extraction (remove the highest priority item) operations are almost same.

A. Array

B. Linked List

C. Heap Data Structures like Binary Heap, Fibonacci Heap

D. None of the above

Answer: Option **C**

Question 5

In a queue, the initial values of front pointer f rare pointer r should be …….. and …….. respectively.

A**.** -1, -1

B. 0, -1

C. 0, 0

D. -1, 0

Answer: Option **B**

Question 6

When new data are to be inserted into a data structure, but there is not available space; this situation is usually called …….

A. Memory Leak

B. Memory Full

C. OverLeak

D. Overflow

Answer: Option **D**

Question 7

Which one of the following is an application of Queue Data Structure?

A. When a resource is shared among multiple consumers.

B. When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes

C. Load Balancing

D. All the above

Answer: Option **D**

Explanation: Queue has many applications like – 1) When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling. 2) When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

Question 8

Priya has a box that looks like a stack and she does the following operations on empty box

PUSH(8)

PUSH(7)

POP

PUSH(1)

PUSH(3)

A. 31_8

B. 8_1_7
C. 8_1_
D. None of these
Answer: Option **C**

Question 9
Which of the following is true about linked list implementation of stack?
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 end, then in pop operation, nodes must be removed from the beginning.
C. Both of the above
D. None of the above
Answer: Option D
Explanation: To keep the Last In First Out order, a stack can be implemented using linked list in two ways:
a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from beginning.
b) In push operation, if new nodes are inserted at the end of linked list, then in pop operation, nodes must be removed from end.

Question 10
Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are
A. Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT
B. Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR
C. Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT
D. Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT
Answer: Option A
Explanation:
Suppose we start filling the queue. Let the maxQueueSize (Capacity of the Queue) is 4.
So the size of the array which is used to implement this circular queue is 5, which is n.
In the begining when the queue is empty, FRONT and REAR point to 0 index in the array.
REAR represents insertion at the REAR index.
FRONT represents deletion from the FRONT index.
enqueue("a"); REAR = (REAR+1)%5; (FRONT = 0, REAR = 1)
enqueue("b"); REAR = (REAR+1)%5; (FRONT = 0, REAR = 2)

enqueue("c"); REAR = (REAR+1)%5; (FRONT = 0, REAR = 3)
enqueue("d"); REAR = (REAR+1)%5; (FRONT = 0, REAR = 4)
Now the queue size is 4 which is equal to the maxQueueSize.
Hence overflow condition is reached.
Now, we can check for the conditions.
When Queue Full :(REAR+1)%n = (4+1)%5 = 0
FRONT is also 0.Hence (REAR + 1) %n is equal to FRONT.
When Queue Empty :REAR was equal to FRONT when empty (because in the starting before filling the queue FRONT = REAR = 0)
Hence Option A is correct.