Data Structures Test Bank 2015

Data Structures Queue

Question 1

Following is C like pseudo code of a function that takes a Queue as an argument, and uses a stack S to do processing.

```
void fun(Queue *Q)
{
    Stack S;  // Say it creates an empty stack S

    // Run while Q is not empty
    while (!isEmpty(Q))
    {
        // deQueue an item from Q and push the dequeued item to S
        push(&S, deQueue(Q));
    }

    // Run while Stack S is not empty
    while (!isEmpty(&S))
    {
      // Pop an item from S and enqueue the poppped item to Q
      enQueue(Q, pop(&S));
    }
}
```

What does the above function do in general?

A Removes the last from Q

B Keeps the Q same as it was before the call

C Makes Q empty

D Reverses the Q

Question 1 Explanation:

The function takes a queue Q as an argument. It dequeues all items of Q and pushes them to a stack S. Then pops all items of S and enqueues the items back to Q. Since stack is LIFO order, all items of queue are reversed.

Data Structures Test Bank 2015

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

Question 2 Explanation:
See http://www.geeksforgeeks.org/applications-of-queue-data-structure/ for details.

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

Question 3 Explanation:
A queue can be implemented using two stacks.

Data Structures Test Bank 2015

Question 4

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

A 1

B 2

C 3

D 4

Question 4 Explanation:
A stack can be implemented using two queues.

.

Question 5

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

Question 5 Explanation:
See http://en.wikipedia.org/wiki/Priority_queue

Question 6

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 end, then in pop operation, nodes must be removed from the beginning.

C Both of the above

D None of the above

Question 6 Explanation:
To keep the **F**irst **I**n **F**irst **O**ut order, a queue can be implemented using linked list in any of the given two ways.

Question 7

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

Data Structures Test Bank 2015

Question 7 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.
```

Question 8

A Priority-Queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is given below: 10, 8, 5, 3, 2 Two new elements ”1‘ and ”7‘ are inserted in the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

A 10, 8, 7, 5, 3, 2, 1

B 10, 8, 7, 2, 3, 1, 5

C 10, 8, 7, 1, 2, 3, 5

D 10, 8, 7, 3, 2, 1, 5

Question 8 Explanation:
See question 4 of http://www.geeksforgeeks.org/data-structures-and-algorithms-set-22/

Question 9

An implementation of a queue Q, using two stacks S1 and S2, is given below:

```
void insert(Q, x) {
   push (S1, x);
}
void delete(Q){
   if(stack-empty(S2)) then
      if(stack-empty(S1)) then {
          print(“Q is empty”);
          return;
      }
      else while (!(stack-empty(S1))){
          x=pop(S1);
          push(S2,x);
      }
   x=pop(S2);
}
```

Let n insert and m (<=n) delete operations be performed in an arbitrary order on an empty queue Q. Let x and y be the number of push and pop operations performed respectively in the process. Which one of the following is true for all m and n?

A n+m <= x < 2n and 2m <= y <= n+m

B n+m <= x < 2n and 2m<= y <= 2n

C 2m <= x < 2n and 2m <= y <= n+m

D 2m <= x <2n and 2m <= y <= 2n

Question 9 Explanation:
The order in which insert and delete operations are performed matters here. The best case: Insert and delete operations are performed alternatively. In every delete operation, 2 pop and 1 push operations are performed. So, total m+ n push (n push for insert() and m push for delete()) operations and 2m pop operations are performed. The worst case: First n elements are inserted and then m elements are deleted. In first delete operation, n + 1 pop operations and n push operation are performed. Other than first, in all delete operations, 1 pop operation is performed. So, total m + n pop operations and 2n push operations are performed (n push for insert() and n push for delete())

Question 10

Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q){
   m = k
   while (Q is not empty and m  > 0) {
      Dequeue(Q)
      m = m - 1
   }
}
```

What is the worst case time complexity of a sequence of n MultiDequeue() operations on an initially empty queue? (GATE CS 2013) (A) (B) (C) (D)

A A

B B

C C

D D

Question 10 Explanation:
Since the queue is empty initially, the condition of while loop never becomes true. So the time complexity is [Tex]\Theta(n)[/Tex].

Question 11

Consider the following pseudo code. Assume that IntQueue is an integer queue. What does the function fun do?

```
void fun(int n)
{
    IntQueue q = new IntQueue();
    q.enqueue(0);
    q.enqueue(1);
    for (int i = 0; i < n; i++)
    {
        int a = q.dequeue();
        int b = q.dequeue();
        q.enqueue(b);
        q.enqueue(a + b);
        ptint(a);
    }
}
```

A Prints numbers from 0 to n-1

B Prints numbers from n-1 to 0

C Prints first n Fibonacci numbers

D Prints first n Fibonacci numbers in reverse order.

Question 11 Explanation:
The function prints first n Fibonacci Numbers. Note that 0 and 1 are initially there in q. In every iteration of loop sum of the two queue items is enqueued and the front item is dequeued.

Question 12

Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q){
   m = k
   while (Q is not empty and m  > 0) {
      Dequeue(Q)
      m = m - 1
   }
}
```

What is the worst case time complexity of a sequence of n MultiDequeue() operations on an initially empty queue? (GATE CS 2013)

```
(A)

(B)

(C)

(D)
```

A A

B B

C C

D D

Data Structures Test Bank 2015

Question 13

Suppose implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?

A A queue cannot be implemented using this stack.

B A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.

C A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.

D A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

Question 13 Explanation:
To DEQUEUE an item, simply POP. To ENQUEUE an item, we can do following 3 operations 1) REVERSE 2) PUSH 3) REVERSE