Data Structures and Algorithms- Quiz 3

Question 1

1 / 1 pts

What is the need for a circular queue?

● Effective usage of memory

○ Easier computations

○ To delete elements based on priority

○ Implement LIFO principle in queues

Question 2

1 / 1 pts

Consider a circular queue of capacity (N-1) elements is implemented with an array of N elements. We are assuming that the insertion and deletion operations are carried out with using REAR and FRONT as array index variables, respectively. Initially REAR=FRONT=0, then the condition to detect the queue full and queue empty

● Full: (REAR+1)%N=FRONT and Empty: REAR==FRONT

○ Full: (REAR+1)%N=FRONT and Empty: REAR==(FRONT+1)%N

○ Full: REAR=FRONT and Empty: REAR==(REAR+1)%N

○ Full: REAR==FRONTand Empty: (REAR+1)%N==FRONT

Question 3

1 / 1 pts

Consider the following statements:

(a) First-in-first out types of computations are efficiently supported by STACKS.

(b) Implementing LISTS on linked lists is more efficient than implementing LISTS on an array for almost all the basic LIST operations.

(c) Implementing QUEUES on a circular array is more efficient than implementing QUEUES on a linear array with two indices.

(d) Last-in-first-out type of computations are efficiently supported by QUEUES.

● b and c are correct

○ a and b

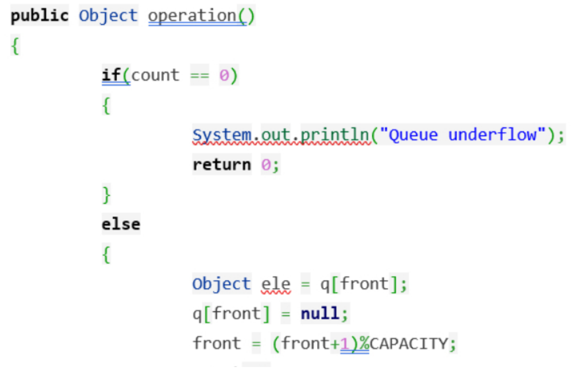
○ c and d

○ a and d

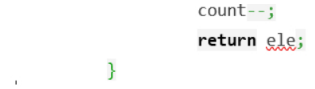
Question 4

1 / 1 pts

Specify the operation(count is the number of elements in the queue)



```
public Object operation()
{
        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;
        }
```

● dequeue

○ enqueue

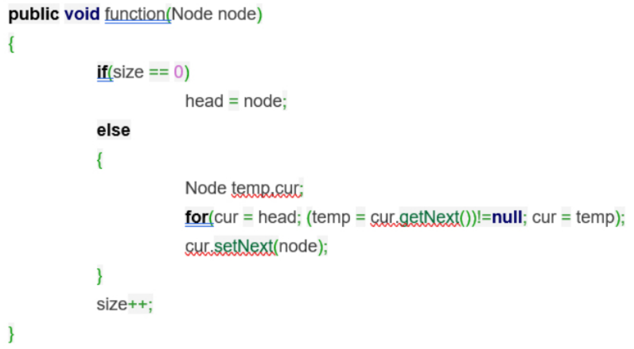
○ pop

○ push

Question 5

1 / 1 pts

What is the functionality of the following code?



```
public void function(Node node)
{
        if(size == 0)
                head = node;
        else
        {
                Node temp,cur;
                for(cur = head; (temp = cur.getNext())!=null; cur = temp);
                cur.setNext(node);
        }
        size++;
}
```

● Inserting a node at the end of the list

○ Inserting a node at the beginning of the list

○ Deleting a node at the beginning of the list

○ Deleting a node at the end of the list

Question 6

1 / 1 pts

The number of edges from the root to the node is called _________ of that node.

● Depth

○ Length

○ Width

○ Height

Question 7

1 / 1 pts

The number of edges from the node to the deepest leaf is called ________ of the tree.

● height

○ depth

○ length

○ width

Question 8

1 / 1 pts

Question 8

Linked list is considered as an example of ___________ type of memory allocation

● Dynamic

○ Static

○ Compile time

○ Heap

Question 9

1 / 1 pts

What does the following function do for a given Linked List with first node as head?

```
void fun(struct node* head)
{
  if(head == NULL)
  return;
  fun(head->next);
  printf("%d  ", head->data);
}
```

● Prints all nodes of linked list in reverse order

○ Prints all nodes of linked lists

○ Prints alternate nodes of Linked List

○ Prints alternate nodes in reverse order

Question 10

1 / 1 pts

In a full binary tree if there are L leaves, then total number of nodes N are?

● N = 2*L – 1

○ N = 2*L

○ N = L + 1

○ N = L – 1