"Circular Linked List".

1. What differentiates a circular linked list from a normal linked list?
a) You cannot have the 'next' pointer point to null in a circular linked list
b) It is faster to traverse the circular linked list
c) You may or may not have the 'next' pointer point to null in a circular linked list
d) Head node is known in circular linked list
View Answer
Answer: c
Explanation: The 'next' pointer points to null only when the list is empty, otherwise it points to the head of the list. Every node in a circular linked list can be a starting point(head).

2. How do you count the number of elements in the circular linked list?
a)

```
public int length(Node head)
{
         int length = 0;
         if( head == null)
                  return 0;
         Node temp = head.getNext();
         while(temp != head)
         {
                  temp = temp.getNext();
                  length++;
         }
         return length;
}
```

b)

```
public int length(Node head)
{
         int length = 0;
         if( head == null)
                  return 0;
         Node temp = head.getNext();
         while(temp != null)
```

```
        {
                temp = temp.getNext();
                length++;
        }
        return length;
}
```
c)

Take Data Structure I Tests Now!

```
public int length(Node head)
{
        int length = 0;
        if( head == null)
                return 0;
        Node temp = head.getNext();
        while(temp != head && temp != null)
        {
                temp = head.getNext();
                length++;
        }
        return length;
}
```
d)

```
public int length(Node head)
{
        int length = 0;
        if( head == null)
                return 0;
        Node temp = head.getNext();
        while(temp != head && temp == null)
        {
                temp = head.getNext();
                length++;
        }
        return length;
}
```
View Answer

3. What is the functionality of the following piece of code? Select the most appropriate.

```
public void function(int data)
{
         int flag = 0;
         if( head != null)
         {
                  Node temp = head.getNext();
                  while((temp != head) && (!(temp.getItem() == data)))
                  {
                           temp = temp.getNext();
                           flag = 1;
                           break;
                  }
         }
         if(flag)
                  System.out.println("success");
         else
                  System.out.println("fail");
}
```

a) Print success if a particular element is not found
b) Print fail if a particular element is not found
c) Print success if a particular element is equal to 1
d) Print fail if the list is empty

View Answer

Answer: b
Explanation: The function prints fail if the given element is not found. Note that this option is inclusive of option “Print fail if the list is empty”, the list being empty is one of the cases covered.

4. What is the time complexity of searching for an element in a circular linked list?
a) O(n)
b) O(nlogn)
c) O(1)
d) O(n^2)

View Answer

Answer: a
Explanation: In the worst case, you have to traverse through the entire list of n elements.

5. Which of the following application makes use of a circular linked list?
a) Undo operation in a text editor
b) Recursive function calls
c) Allocating CPU to resources

d) Implement Hash Tables
View Answer
Answer: c
Explanation: Generally, round robin fashion is employed to allocate CPU time to resources which makes use of the circular linked list data structure. Recursive function calls use stack data structure. Undo Operation in text editor uses doubly linked lists. Hash tables uses singly linked lists.

6. Choose the code snippet which inserts a node to the head of the list?
a)

```
public void insertHead(int data)
{
         Node temp = new Node(data);
         Node cur = head;
         while(cur.getNext() != head)
                  cur = cur.getNext()
         if(head == null)
         {
                  head = temp;
                  head.setNext(head);
         }
         else
         {
                  temp.setNext(head);
                  head = temp;
                  cur.setNext(temp);
         }
         size++;
}
```

b)

```
public void insertHead(int data)
{
         Node temp = new Node(data);
         while(cur != head)
                  cur = cur.getNext()
         if(head == null)
         {
                  head = temp;
                  head.setNext(head);
         }
```

```
        else
        {
                temp.setNext(head.getNext());
                cur.setNext(temp);
        }
        size++;
}
```

c)

```
public void insertHead(int data)
{
        Node temp = new Node(data);
        if(head == null)
        {
                head = temp;
                head.setNext(head);
        }
        else
        {
                temp.setNext(head.getNext());
                head = temp;
        }
        size++;
}
```

d)

```
public void insertHead(int data)
{
        Node temp = new Node(data);
        if(head == null)
        {
                head = temp;
                head.setNext(head.getNext());
        }
        else
        {
                temp.setNext(head.getNext());
                head = temp;
        }
        size++;
}
```

View Answer
Answer: a
Explanation: If the list is empty make the new node as ‘head’, otherwise traverse the list to the end and make its ‘next’ pointer point to the new node, set the new node’s next point to the current head and make the new node as the head.

7. What is the functionality of the following code? Choose the most appropriate answer.

```
public int function()
{
         if(head == null)
                  return Integer.MIN_VALUE;
         int var;
         Node temp = head;
         while(temp.getNext() != head)
                  temp = temp.getNext();
         if(temp == head)
         {
                  var = head.getItem();
                  head = null;
                  return var;
         }
         temp.setNext(head.getNext());
         var = head.getItem();
         head = head.getNext();
         return var;
}
```

a) Return data from the end of the list
b) Returns the data and deletes the node at the end of the list
c) Returns the data from the beginning of the list
d) Returns the data and deletes the node from the beginning of the list
View Answer
Answer: d
Explanation: First traverse through the list to find the end node, then manipulate the ‘next’ pointer such that it points to the current head’s next node, return the data stored in head and make this next node as the head.

8. What is the functionality of the following code? Choose the most appropriate answer.

```
public int function()
```

```
{
         if(head == null)
                  return Integer.MIN_VALUE;
         int var;
         Node temp = head;
         Node cur;
         while(temp.getNext() != head)
         {
                  cur = temp;
                  temp = temp.getNext();
         }
         if(temp == head)
         {
                  var = head.getItem();
                  head = null;
                  return var;
         }
         var = temp.getItem();
         cur.setNext(head);
         return var;
}
```

a) Return data from the end of the list
b) Returns the data and deletes the node at the end of the list
c) Returns the data from the beginning of the list
d) Returns the data and deletes the node from the beginning of the list
View Answer
Answer: b
Explanation: First traverse through the list to find the end node, also have a trailing pointer to find the penultimate node, make this trailing pointer's 'next' point to the head and return the data stored in the 'temp' node.

9. Which of the following is false about a circular linked list?
a) Every node has a successor
b) Time complexity of inserting a new node at the head of the list is O(1)
c) Time complexity for deleting the last node is O(n)
d) We can traverse the whole circular linked list by starting from any point
View Answer
Answer: b
Explanation: Time complexity of inserting a new node at the head of the list is O(n) because you have to traverse through the list to find the tail node.

10. Consider a small circular linked list. How to detect the presence of cycles in this list effectively?
a) Keep one node as head and traverse another temp node till the end to check if its 'next points to head
b) Have fast and slow pointers with the fast pointer advancing two nodes at a time and slow pointer advancing by one node at a time
c) Cannot determine, you have to pre-define if the list contains cycles
d) Circular linked list itself represents a cycle. So no new cycles cannot be generated
View Answer
Answer: b
Explanation: Advance the pointers in such a way that the fast pointer advances two nodes at a time and slow pointer advances one node at a time and check to see if at any given instant of time if the fast pointer points to slow pointer or if the fast pointer's 'next' points to the slow pointer. This is applicable for smaller lists.