

ADS CCEE Practice Quiz

Total points 12/20 ?

The respondent's email (prathameshpatkar890@gmail.com) was recorded on submission of this form.

0 of 0 points

12 Digit PRN *

240840320073

Full Name *

Prathamesh Patkar

Centre *

Kharghar ▼

Email Address *

prathameshpatkar890@gmail.com

Untitled Section

12 of 20 points

✓ If the size of the array used to implement a circular queue is MAX_SIZE. *1/1
How rear moves to traverse inorder to insert an element in the queue?

☐ $\text{rear} = (\text{rear} \% 1) + \text{MAX_SIZE}$

☒ $\text{rear} = (\text{rear} + 1) \% \text{MAX_SIZE}$ ✓

☐ $\text{rear} = \text{rear} + (1 \% \text{MAX_SIZE})$

☐ $\text{rear} = \text{rear} \% (\text{MAX_SIZE} + 1)$

✓ In worst case, the number of comparison need to search a singly linked *1/1
list of length n for a given element is

☐ $\log n$

☐ $n/2$

☐ $\log_2 n - 1$

☒ n ✓

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

*1/1

```
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;
}
```

- ☐ Return data from the end of the list
- ☐ Returns the data and deletes the node at the end of the list
- ☐ Returns the data from the beginning of the list
- ☒ Returns the data and deletes the node from the beginning of the list ✓

✗ The concatenation of two list can performed in $O(1)$ time. Which of the following variation of linked list can be used? *0/1

- ☐ Singly linked list
- ☒ Doubly linked list
- ☐ Circular doubly linked list
- ☐ Array implementation of list

Correct answer

- ☒ Circular doubly linked list

✗ A circularly linked list is used to represent a Queue. A single variable p is used to access the Queue. To which node should p point such that both the operations enqueue and dequeue can be performed in constant time? *0/1

- ☐ rear node
- ☐ front node
- ☐ not possible with a single pointer
- ☒ node next to front

Correct answer

- ☒ rear node



*1/1

```
public class Test
{
    public static void main(String[] args) {
        String str = null;
        System.out.println(str.valueOf(10));
    }
}
```

Which of the following statements correctly describes the behavior of this program?

- ☐ This program will result in a compiler error.
- ☐ This program will throw a NullPointerException.
- ☒ This program will print 10 in the console.
- ☐ This program will print null in the console.



✗ What is the functionality of the following piece of code? *

0/1

```
public void function(Object item)
{
    Node temp=new Node(item,trail);
    if(isEmpty())
    {
        head.setNext(temp);
        temp.setNext(trail);
    }
    else
    {
        Node cur=head.getNext();    while(cur.getNext()!=trail)
        {
            cur=cur.getNext();
        }
        cur.setNext(temp);
    }
    size++;
}
```

- ☐ Insert at the front end of the dequeue
- ☐ Insert at the rear end of the dequeue
- ☒ Fetch the element at the rear end of the dequeue ✗
- ☐ Fetch the element at the front end of the dequeue

Correct answer

- ☒ Insert at the rear end of the dequeue

✗ The minimum number of fields with each node of doubly linked list is * 0/1

☐ 1

☒ 2

✗

☐ 3

☐ 4

Correct answer

☒ 3

✗ Here is an infix expression: $4 + 3 * (6 * 3 - 12)$. Suppose that we are using the usual stack algorithm to convert the expression from infix to postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression? *0/1

☐ 1

☒ 2

✗

☐ 3

☐ 4

Correct answer

☒ 4

✗ What is the result of the following code? *

0/1

```
String s1 = "Java"; String s2 = "Java";  
StringBuilder sb1 = new StringBuilder();  
sb1.append("Ja").append("va");  
System.out.println(s1 == s2);  
System.out.println(s1.equals(s2));  
System.out.println(sb1.toString() == s1);  
System.out.println(sb1.toString().equals(s1));
```

- ☐ true is printed out exactly once.
- ☒ true is printed out exactly twice. ✗
- ☐ true is printed out exactly three times.
- ☐ true is printed out exactly four times.

Correct answer

- ☒ true is printed out exactly three times.

✗ The following three are known to be the preorder, inorder and postorder sequences of a binary tree. But it is not known which is which. *0/1

MBCAFHPYK

KAMCBYPFH

MABCKYFPH

Pick the true statement from the following.

- ☐ I and II are preorder and inorder sequences, respectively
- ☐ I and III are preorder and postorder sequences, respectively
- ☒ II is the inorder sequence, but nothing more can be said about the other two sequences ✗
- ☐ II and III are the preorder and inorder sequences, respectively

Correct answer

- ☒ II and III are the preorder and inorder sequences, respectively

- ✓ What are the main applications of tree data structure? * 1/1
- Manipulate hierarchical data
 - Make information easy to search
 - Manipulate sorted lists of data
 - Router algorithms
 - Form of a multi-stage decision-making, like Chess Game.
 - As a workflow for compositing digital images for visual effects
- ☐ 1, 2, 3, 4 and 6
- ☐ 1, 2, 3, 4 and 5
- ☐ 1, 3, 4, 5 and 6
- ☒ 1, 2, 3, 4, 5 and 6 ✓

- ✓ Consider the following doubly linked list: head-1-2-3-4-5-tail. What will be the list after performing the given sequence of operations? *1/1
- ```
Node temp = new Node(6,head,head.getNext());
Node temp1 = new Node(0,tail.getPrev(),tail); head.setNext(temp);
temp.getNext().setPrev(temp);
tail.setPrev(temp1);
temp1.getPrev().setNext(temp1);
```
- ☐ head-0-1-2-3-4-5-6-tail
- ☐ head-1-2-3-4-5-6-tail
- ☒ head-6-1-2-3-4-5-0-tail ✓
- ☐ head-0-1-2-3-4-5-tail

✓ What is the functionality of the following piece of code? \*

1/1

```
public int function(int data) {
 Node temp = head;
 int var = 0;
 while(temp != null) {
 if(temp.getData() == data) {
 return var;
 }
 var = var+1;
 temp = temp.getNext();
 }
 return Integer.MIN_VALUE;
}
```

- ☐ Find and delete a given element in the list
- ☐ Find and return the given element in the list
- ☒ Find and return the position of the given element in the list ✓
- ☐ Find and insert a new element in the list

✗ Consider a binary tree with  $n$  nodes, where each node can have at most two children. The height of the tree is defined as the maximum number of edges between the root node and any leaf node. Which of the following statements is true regarding the height  $h$  of this binary tree? \*0/1

- ☐ The height of the tree is always equal to  $n-1$ .
- ☐ The height of the tree can be greater than or equal to  $n-1$ .
- ☐ The height of the tree is always equal to  $\log_2(n)$ .
- ☒ The height of the tree can be greater than or equal to  $\log_2(n)$ . ✗

Correct answer

- ☒ The height of the tree can be greater than or equal to  $n-1$ .

- ✓ Consider a single linked list where F and L are pointers to the first and last elements respectively of the linked list. The time for performing which of the given operations depends on the length of the linked list? \*1/1

F->1->2->3->L

- ☐ Delete the first element of the list
- ☐ Interchange the first two elements of the list
- ☒ Delete the last element of the list ✓
- ☐ Add an element at the end of the list

- ✓ Consider the following operation performed on a stack of size 5. \*1/1

```
Push(1);
Pop();
Push(2);
Push(3);
Pop();
Push(4);
Pop();
Pop();
Push(5);
```

After the completion of all operation, the number of elements present in stack is?

- ☒ 1 ✓
- ☐ 2
- ☐ 3
- ☐ 4

✓ What is the functionality of the following piece of code? \*

1/1

```
public class Test
{
 public static void main(String[] args) {
 String str = null;
 switch (str) { // #1
 case "null":
 System.out.println("null string"); // #2
 break;
 }
 }
}
```

- ☐ This program results in a compiler error in statement #1.
- ☐ This program results in a compiler error in statement #2.
- ☒ This program results in throwing a NullPointerException. ✓
- ☐ This program prints the following: null string.

✓ What is the functionality of the following piece of code? \*

1/1

```
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 beginning of the list
- ☐ Deleting a node at the beginning of the list
- ☒ Inserting a node at the end of the list
- ☐ Deleting a node at the end of the list



- ✓ What is the functionality of the following piece of code? Select the most appropriate. \*1/1

```
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"); }
```

- ☐ Print success if a particular element is not found
- ☒ Print fail if a particular element is not found ✓
- ☐ Print success if a particular element is equal to 1
- ☐ Print fail if the list is empty

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#).

Google Forms

