**Question 1**

Given the following code. How many elements the testSet will contain and in which order?

**public** **class** TestSet {

**enum** vals {

*ONE*, *TWO*, *THREE*

};

**public** **static** **void** main(String args[]) {

Collection coll = **new** ArrayList();

coll.add(vals.*THREE*);

coll.add(vals.*THREE*);

coll.add(vals.*THREE*);

coll.add(vals.*TWO*);

coll.add(vals.*TWO*);

coll.add(vals.*ONE*);

Set testSet = **new** HashSet(coll);

}

}

**Answer:** 3 elements, and the order is not preserved.

**Question 2**

A monitor called mon has 5 threads in its waiting pool; all these waiting threads have the same priority. One of the threads is thread1. How can you notify thread1 so that  it alone moves from Waiting state to Ready State?

**Answer:** You cannot specify which thread will get notified. When you call notify() on a monitor, you have no control over which waiting thread gets notified.

**Question 3**

What is the output of this code fragment?int x=3;

int y =10;

System.out.println(y%x);

**Answer:** Dividing 10 by 3 gives 3 reminder 1, and this 1 forms the result of the modulo expression.

**Question 4**

The constructor for the Math class is private, so it cannot be instantiated. True/False?

**Answer:** True

**Question 5**

Whether the following code will be executed? If it is executed, **what will be the output of the code?**

**public** **class** Threads3 **implements** Runnable {

**public** **void** run() {

System.*out*.print("running");

}

**public** **static** **void** main(String[] args) {

Thread t = **new** Thread(**new** Threads3());

t.run();

t.run();

t.start();

}

}

**Answer:** The code executes and prints "runningrunningrunning".

**Question 6**

What modifiers may be used with an inner class that is a member of an outer class?

**Answer:** The (non-local) inner class may be declared as public, protected, private, static, final, or abstract.

**Question 7**

What is the difference between the Boolean & operator and the && operator?

**Answer:**

The single ampersand is used to perform ‘bit-wise AND’ operation on integer arguments. It constitutes the logical AND operator, which returns 1 if both bits in first and second arguments. Otherwise it returns 0. It always evaluates both the arguments.

A double ampersand is used to perform boolean operations, such as ‘logical AND’. It returns true if both the operands are true, otherwise returns false. It will evaluate the second argument if the first argument is true; otherwise the operation of the second operand is skipped

**Question 8**

If a method is declared as protected, where may the method be accessed?

**Anwer:** The “protected” method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.

**Question 9**

When is the finally clause of a try-catch-finally statement executed?

**Answer:**

The finally clause of the try-catch-finally statement is always executed unless the thread of execution terminates or an exception occurs within the execution of the finally clause.

**Question 10**

What modifiers are allowed for methods in an Interface?

**Answer:**

Only public and abstract modifiers are allowed for methods in interfaces.

**Question 11**

What is the difference between <jsp:include page = ... > and <%@ include file = ... >?

**Answer:** Both the tag includes the information from one page in another. The differences are as follows:  
<jsp:include page = ... >: This is like a function call from one jsp to another jsp. It is executed ( the included page is executed  and the generated html content is included in the content of calling jsp) each time the client page is accessed by the client. This approach is useful for modularizing the web [application](http://www.roseindia.net/interviewquestions/jsp-interview-questions.shtml). If the included file changed then the new content will be included in the output.   
  
<%@ include file = ... >: In this case the content of the included file is textually embedded in the page that have <%@ include file=".."> directive. In this case, if the included file changes, the changed content will not included in the output. This approach is used when the code from one jsp file required to include in multiple jsp files.

**Question 12**

What is the difference between <jsp:forward page = ... > and response.sendRedirect(url) ?

**Answer:** The <jsp:forward> element forwards the request object containing the client request information from one JSP file to another file. The target file can be an HTML file, another JSP file, or a servlet, as long as it is in the same application context as the forwarding JSP file.   
sendRedirect sends HTTP temporary redirect response to the browser, and [browser](http://www.roseindia.net/interviewquestions/jsp-interview-questions.shtml) creates a new request to go the redirected page. The  response.sendRedirect kills the session variables.

**Question 13**

What are all the different scope values for the <jsp:useBean> tag?

**Answer:** <jsp:useBean> tag is used to use any java object in the jsp page. Here are the scope values for <jsp:useBean> tag:  
a) page  
b) request  
c) session and  
d) application

**Question 14**

How you will handle the runtime exception in your jsp page?

**Answer:** The errorPage attribute of the page directive can be used to catch run-time exceptions automatically and then forwarded to an error processing page.

For example:  
<%@ page errorPage="customerror.jsp" %>

**Question 15**

Outline the major difference between the session and cookie?

**Answer:**

Sessions are always stored in the server side whereas cookies are always stored in the client side.

**Question 16**

Explain Quick Sort with code snippet & its complexity?

**Answer:**

Complexity: Θ(n log(n))

It works recursively by a divide-and-conquer strategy.

void quicksort (int[] a, int lo, int hi)

{

// lo is the lower index, hi is the upper index

// of the region of array a that is to be sorted

int i=lo, j=hi, h;

// comparison element x

int x=a[(lo+hi)/2];

// partition

do

{

while (a[i]<x) i++;

while (a[j]>x) j--;

if (i<=j)

{

h=a[i]; a[i]=a[j]; a[j]=h;

i++; j--;

}

} while (i<=j);

// recursion

if (lo<j) quicksort(a, lo, j);

if (i<hi) quicksort(a, i, hi);

}

**Question 17**

Explain bubble sort with code snippet and what is its complexity?

Complexity: O(n2)

public void bubbleSort(int[] arr) {

boolean swapped = true;

int j = 0;

int tmp;

while (swapped) {

swapped = false;

j++;

for (int i = 0; i < arr.length - j; i++) {

if (arr[i] > arr[i + 1]) {

tmp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = tmp;

swapped = true;

}

}

}

}

**Question 18**

Write an algorithm to find the factorial of a given number?

public static long factorial(long n) {

if (n <= 1)

return 1;

else

return n\*factorial(n-1);

}

**Question 19**

Write an algorithm to find whether the given string is a palindrome?

Public class palindrome  
{  
Boolean isPalindrome(String testString){  
    return(testString.Equals(reverse(testString)));  
}  
  
public static String reverse(String s){  
    String and;  
      
    if (s.Length() <= 1)  
        return s;  
      
    else{     
        char lastC = s.CharAt(s.Length()-1);          
        String stringLeft = s.Substring(0, s.Length() -1);  
        return and = lastC + reverse(stringLeft);     
        }  
    }  
}

**Question 20**

Write an iterative function to find the position of the element in an array using binary search?

**public** **static** **int** binarySearch( Comparable [ ] a, Comparable x )

{

**int** low = 0;

**int** high = a.length - 1;

**int** mid;

**while**( low <= high )

{

mid = ( low + high ) / 2;

**if**( a[ mid ].compareTo( x ) < 0 )

low = mid + 1;

**else** **if**( a[ mid ].compareTo( x ) > 0 )

high = mid - 1;

**else**

**return** mid;

}

**return** NOT\_FOUND;

}