

Sample Multiple Choice Question #1

(A)

This question with a display of **x** and **y**, which is **10 20**.

Method **swap** is called after the first output display.

The values of **p** and **q** are exchanged.

This time there is no output display inside the **swap** method.

The second display of **x** and **y**, in the **main** method, displays the same **10 20** values.

Remember that a change in *formal* (receiving) parameters does not alter *actual* (calling) parameters.

Three more questions follow with the **swap** method to help illustrate this point.

Sample Multiple Choice Question #2

(A)

Segment 1 correctly uses the **for..each** loop to compute the sum of the numbers.

Segment 2 does not use the **for..each** loop correctly. **item** is an element and not an index.

Segment 3 uses completely wrong syntax for the **for..each** loop.

Sample Multiple Choice Question #3

(D)

Call #	s	n	Method Prints	Method Call
1	TANGO	5	TANGO	method0909("TANG");
2	TANG	4	TANG	method0909("TAN");
3	TAN	3	TAN	method0909("TA");
4	TA	2	TA	method0909("T");
5	T	1	T	method0909("");
6	empty	0		

As the recursive methods displays in LIFO (Last In, First Out) sequence, the **Methods Prints** displays will occur in reverse order.

Sample Multiple Choice Question #4

(C)

The **[400 . . 1600]** range has **1201** different numbers.

Multiply **1201** times **rndDouble** followed by an **(int)** cast gives the range **[0 . . 1200]**.

Add the smallest value **400** to the numbers gives the desired range **[400 . . 1600]**.

Sample Multiple Choice Question #5

(C)

The question indicates that two processes are required.

It is also states that quick access to student records occurs frequently.

A second requirement is an end-of-semester access GPA list.

Method 2 sorts the data, such that students are quickly accessed for updating.

Sorting once per semester is infrequent enough to make that a second priority.

Sample Multiple Choice Question #6

(A)

Operation	A	B	C	Stack s	Queue q	Output
	1	2	3	Empty	Empty	
s.push(a)				1		
s.push(b)				2, 1		
s.push(c)				3, 2, 1		
q.add(s.pop)				2, 1	3	
q.add(s.pop)				1	3, 2	
q.add(s.pop)				Empty	3, 2, 1	
q.add(c)					3, 2, 1, 3	
q.add(b)					3, 2, 1, 3, 2	
q.add(a)					3, 2, 1, 3, 2, 1	
OUTPUT						3 2 1 3 2 1

Sample Multiple Choice Question #7

(C)

The **PriorityQueue** object stores **String** values.

The **String** class implements method **compareTo** alphabetically.

This means that elements from the **pqueue** object are removed in alphabetical order.

Sample Multiple Choice Question #8

. (E)

This problem requires that you start with all the nodes and subtract the root and all the leaves.

The number of leaves in a full binary tree with n levels is $2^{(n-1)}$.

The number of roots is 1 .

The number of total nodes in a full binary tree is $2^n - 1$.

The formula becomes $(2^n - 1) - (2^{(n-1)} - 1)$, which simplifies to $2^n - 2^{(n-1)} - 2$.

Sample Multiple Choice Question #9

. (E)

Implementation 1 is correct.

Implementation 2 is correct.

Implementation 3 is correct.

The **toString** method is redefined for **Set** classes.

The **for..each** loop is ideal for **Set** object traversal.

The **Iterator** object can traverse **Set** objects.

Sample Multiple Choice Question #10

. (A)

The **put** method first returns the current value found with the provided key.
The second step is to replace the current value with the newly provided value.
Question 10 prints the output of the **put** statements, which are the original values.

Sample Free Response Question A-Exam 2

Question 2.

Part (a).

```
public int nextInteger(int n)
{
    double temp = getRandom();
    temp = temp * n;
    return (int) temp;
}
```

Part (b).

```
public Random2(double seed)
{
    super(seed);
}
```

Part (c).

```
public int nextInteger(int start, int end)
{
    int range = end - start + 1;
    int temp = nextInteger(range);
    return temp + start;
}
```