UNIT 1
Primitive Types

16. Consider this mathematical expression.

$$p = \frac{2a + 3b}{3a - 2b}$$

Which of the following Java program statements are correct to use in a program? Assume that variables p, a and b have been declared.

(A)
```
p = 2a + 3b / 3a - 2b;
```

(B)
```
p = (2a + 3b) / (3a - 2b);
```

(C)
```
p = (2*a + 3*b) / (3*a - 2*b);
```

(D)
```
p = 2*a + 3*b / 3*a - 2*b;
```

(E)
```
p = 2*a + (3*b / 3*a) - 2*b;
```

UNIT 2
Using Objects

30. Consider the following code segment.

```
String str1 = "Queen Mary 2";
String str2 = "Queen Mary 2";
String str3 = new String("Queen Victoria");
String str4 = new String("Queen Victoria");
System.out.println(str1 == str2);
System.out.println(str1.equals(str2));
System.out.println(str3 == str4);
System.out.println(str3.equals(str4));
```

What is printed when the code segment is executed?

(A)	(B)	(C)	(D)	(E)
true true false true	false true false true	true true true true	true true false false	false false true true

UNIT 3
Boolean Expressions and if Statements

25. Consider the following code segment.

```
boolean a = <unknown boolean>
boolean b = <unknown boolean>
boolean x = !(a && b);
boolean y = !a || !b;
boolean z = x == y;
```

What is true about this program segment?

(A) `z` will always be `true`.
(B) `z` will always be `false`.
(C) It is not possible to determine the value of `z`.
(D) The program segment demonstrates *De Morgan's Law*.
(E) Both (A) and (D) are correct.

UNIT 4
Iteration

07. Consider the following code segment.

```
int n = 0;
double sum = 0.0;
while (n < 1000)
{
    int rnd = (int) (Math.random() * 99 + 1);
    sum += rnd;
}
double average = sum / n;
System.out.println(average);
```

The program is meant to compute the average of random numbers in the `[1..99]` range.
Is the program code correct, and which of the following describes the program's behavior correctly?

(A) The program computes the average correctly.
(B) The program does not compile due to a syntax error.
(C) The program does not execute correctly due to a division by zero error.
(D) The program does not execute correctly. It is stuck in an infinite loop.
(E) The program computes the average correctly, but the random numbers are wrong.

UNIT 5
Writing Classes

25. Consider the following code segment and `Fraction` class.

```
Fraction f1 = new Fraction(3,4);
Fraction f2 = new Fraction(2,3);
Fraction f3 = new Fraction(1,1);
f3.multiply(f1,f2);
System.out.println(f3);

class Fraction
{
   private int num;
   private int den;

   public Fraction (int n, int d)
   {
      num = n;
      den = d;
   }

   public void multiply(Fraction f1, Fraction f2)
   {
      this.num = f1.num * f2.num;
      this.den = f1.den * f2.den;
   }

   public String toString()
   {
      return num + "/" + den;
   }
}
```

What is printed when the program is executed?

(A) `1/1`

(B) Syntax error, attempted access at `private` attribute

(C) Runtime error, `ArithmeticException`

(D) `1/2`

(E) `6/12`

13. Consider the following code segment and `getMean` method.

```
int[] list = {12,26,34,40,52,67,77,81,98};
double mean = getMean(list);
System.out.println(mean);

public static double getMean(int[] x)
{
   /* Missing Code */
}
```

Method `getMean` is meant to return the *mean* of the `list` array elements.

Which of the following replacements for **/* Missing Code */** satisfies that requirement?

(A)
```
int sum = 0;
for (int k = 0; k < x.length; k++)
       sum += x[k];
   return sum / x.length;
```

(B)
```
double sum = 0.0;
   for (int item : x)
   sum += item;
return sum / x.length;
```

(C)
```
double sum = 0.0;
   for (int k = 1; k <= x.length; k++)
   sum += x[k];
return sum / x.length;
```

(D)
```
double sum = 0.0;
for (int k = 0; k < x.length; k++)
   sum += x[k];
return sum / x.length;
```

(E) Both (B) and (D)

UNIT 7
ArrayList

16. Consider the following code segment.

```
int[] list1 = {11,22,33,44,55,66,77};
ArrayList<Integer> list2 = new ArrayList<Integer>();
for (int item : list1)
   list2.add(item);
for (int k = 0; k < list2.size(); k++)
   list2.remove(k);
System.out.println(list2);
```

What is printed when the code segment is executed?

(A) `[22, 44, 66]`

(B) `[11, 33, 55, 77]`

(C) `[44, 55, 66, 77]`

(D) `[11]`

(E) `[]`

UNIT 8
2D Array

08. Consider the following code segment, which correctly creates a 2D array.

```
ArrayList<Integer> row1 = new ArrayList<Integer>();
row1.add(100);
row1.add(200);
row1.add(300);

ArrayList<Integer> row2 = new ArrayList<Integer>();
row2.add(400);
row2.add(500);
row2.add(600);

ArrayList<Integer> row3 = new ArrayList<Integer>();
row3.add(700);
row3.add(800);
row3.add(900);

ArrayList<ArrayList<Integer>> matrix = new ArrayList<ArrayList<Integer>>();
matrix.add(row1);
matrix.add(row2);
matrix.add(row3);
```

Which of the following program statements will display `500`?

(A) `System.out.println(matrix.get(get(1));`

(B) `System.out.println(matrix.get(1,1));`

(C) `System.out.println(matrix.get(1).get(1));`

(D) `System.out.println(matrix.get(1),matrix.get(1));`

(E) `System.out.println(matrix.matrix.get(1,1));`

05. Consider the following code segment and three classes.

```
Class1 c1 = new Class1();
Class2 c2 = new Class2();
Class3 c3 = new Class3();

class Class1
{
   public Class1()
   {
      System.out.println("Class1");
   }
}

class Class2 extends Class1
{
   public Class2()
   {
      System.out.println("Class2");
   }
}

class Class3 extends Class2
{
   public Class3()
   {
      System.out.println("Class3");
   }
}
```

What is printed when the program is executed?

(A)	(B)	(C)	(D)	(E)
Class1 Class1 Class2 Class1 Class2 Class3	Class1 Class2 Class3	Class3 Class2 Class1	Class3 Class2 Class1 Class2 Class1 Class1	null null null

UNIT 10
Recursion

09. Consider the following recursive `mystery` method.

```
public static void mystery (int[] list, int q)
{
   if (q < list.length-1)
   {
      if (list[q] > list[q+1])
      {
         int temp = list[q];
         list[q] = list[q+1];
         list[q+1] = temp;
      }
      mystery(list,q+1);
   }
}
```

Assume that method `mystery` is called by `mystery(numbers,0);`
where `numbers` is an array of random numbers.

Which of the following describes the behavior of the `mystery` method?

(A) `mystery` sorts the `list` array in ascending order.

(B) `mystery` places the smallest element of `list` at the start of the `list` array.

(C) `mystery` places the smallest element of `list` at the end of the `list` array.

(D) `mystery` places the largest element of `list` at the start of the `list` array.

(E) `mystery` places the largest element of `list` at the end of the `list` array.

Snooker Questions

02. Consider the following code segment.

```
double a = <Unknown double value>
double b = 3.0 * a;
double c = 4.0 * a;
double result = b / c;
System.out.println(result);
```

What is printed as a result of executing the code segment?

(A) Cannot be determined without knowing the value of a.

(B) `0.0`

(C) `1.25`

(D) Runtime error. ArithmeticException

(E) `0.75`

Snooker Question 02 Explanation Answer: E

The secret of this question is that both variable **b** and **c**, which are known, are both multiplied times **a**, which is not known.

Then the result becomes **b / c**.
Let's do a substitution for **b** and **c**.
This gives us **(3.0 * a) / (4.0 * a).**

This expression can be simplified by dividing both numerator and denominator by the same factor value of **a**, which leaves **3.0 / 4.0** or **0.75**.

It does not matter what the value of **a** starts with. Since **a** is a common factor to reduce the fraction, the result will always be **0.75** with any value of **a**.

Free-Response Samples

Question 1. (Methods and Control Structures)

Over 2000 years ago, Greek mathematician Euclid devised a clever algorithm to compute the Greatest Common Factor (GCF) of two integers. The steps are shown below.

Euclid's Greatest Common Factor Algorithm

Step 1. Find the remainder of dividing nbr1 by nbr2.

Step 2. If the remainder equals 0, You are finished and the GCF is nbr2

Step 3. If the remainder is not 0,
then nbr1 becomes nbr2 and nbr2 becomes the remainder.

Step 3. Repeat the process by returning to step 1.

Part (a).

Write method `getGCF` based on Euclid's Algorithm, shown above.

Complete method `getGCF` below.

```
/** Precondition:  n1 and n2 are positive integers.
 *  Postcondition: returns the GCF of n1 and n2.
 */
public static int getGCF (int n1, int n2)
```

Part (b).

Write method `getLCM` below. You are not provided with an algorithm to compute the Least Common Multiple (LCM). Knowing the GCF of two numbers makes it easier to compute the LCM.

In computing the Least Common Multiple (LCM), you may use method `getGCF` from **Part (a)** whether you wrote the method correctly of not.

Complete method `getLCM` below.

```
/** Precondition:  n1 and n2 are positive integers.
 *                 getGCF is available to compute the LCM of n1 and n2.
 *  Postcondition: getLCM returns the LCM of n1 and n2.
 */
public static int getLCM (int n1, int n2)
```

Question 2. (Class)

This question involves a `Fraction` class. It is a class that can handles rational numbers with the four basic arithmetic operations for fractions (add, subtract, multiply, divide), reduce the result, and then display the three fractions in the following format:

1/2 + 1/3 = 5/6

For this question the `Fraction` will be limited to addition only. The result of the addition needs to be reduced and that can be done with a provided `getGCF` method that returns the Greatest Common Factor (GCF) of two integer attributes of the `Fraction` class. The incomplete `Fraction` class is shown below.

```
class Fraction
{
    public Fraction (int n, int d)
    {
        num = n;
        den = d;
    }

    public void add(Fraction fract1, Fraction fract2)
    {
        /* To be Completed in Part (a) */
    }

    /**  Postcondition: returns the GCF of n1 and n2. */
    public static int getGCF (int n1, int n2)
    {
        /* Can be used in writing method add */
    }

   public String toString()
   {
        /* To be Completed in Part (b) */
   }
}
```

Question 2 Continued.

Part (a).

Method `add` is not a `return` method. It is a `void` method that can be used in a program statement, such as `fract3.add(fract1, fract2);` The results of the addition of objects `fract1` and `fract` 2, which are a numerator (`num`) and a denominator (`den`) need to be stored in the `fract3` object.

Complete method `add` below. This is meant to be rational number addition. It is not meant to be decimal number computation, such as `0.25 + 0.50 = 0.75`. It is meant to be `1/4 + 1/2 = 3/4`.

Method `add` is not a `return` method. It is a `void` method that is to be used in a program statement. The result of adding fractions from object `fract1` and `fract2` are expected be reduced. You may use method `getGCF`, which was described previously.

```
/** Precondition:  fract1 and fract2 are objects with attributes initialized.
 *  Postcondition: the results of adding fract1 to fract is stored in "this".
 */

public void add(Fraction fract1, Fraction fract2)
```

Part (b).

Complete method `toString` below.

```
/**
 * Precondition:  Attributes num and den are initialized, or altered, in the
 *                this Fraction object.
 * Postcondition: returns a String in the num/den format, such as 3/4
 */
 public String toString()
```

Question 3. (Array/ArrayList)

The Standard Deviation, called SD, is an important statistical tool that is used in curving grades and assigning scores to tests, such as the SAT. The image below shows the formula for computing the SD

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

In this formula **X** represents an individual score and **X** with a horizontal bar is the mean of the scores. To compute the SD the following steps are used.

1. Compute the mean of the numbers. (divide sum of the numbers by the number count)
2. Subtract each number from the mean, square it, and add it to a growing sum.
3. Divide the sum by **(n-1)** where **n** is the quantity of numbers.
4. Finally, take the square root of the result from step 3.

Part (a).

For this question you will be writing methods `getMean` and `getSD` to compute the SD. You can assume that a static array of random integers exists already. You will use this static array to compute the mean first and then the Standard Deviation.

Complete method `getMean` below.

```
/** Precondition:  list is a non-empty static array of random integers
 *  Postcondition: returns the mean of the numbers in list.
 */
public static double getMean(int[] list)
```

Part (b).

For part (b) you will write method `getSD`. In writing method `getSD` you may use the `getMean` method from **Part (a)** whether you wrote the method correctly or not. Use the formula and steps shown earlier for computing the standard deviation.

Complete method `getSD` below.

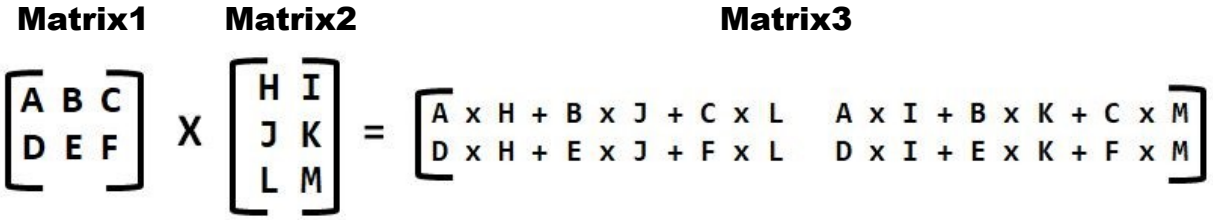
```
/** Precondition:  list is a non-empty static array of random integers
 *  Postcondition: returns the Standard Deviation of the numbers in list.
 */
public static double getStdDev(int[] list)
```

Question 4. (2D Array)

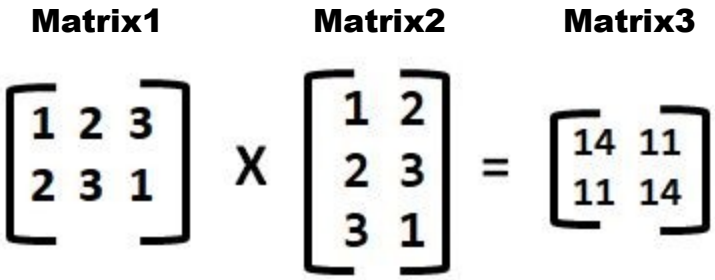
Question is a 2D static array question, which involves matrix multiplication.

Matrix addition and subtraction use simple arithmetic where one element of matrix1 and one corresponding element of matrix2 are added or subtracted and then the sum or difference is placed in the corresponding location of matrix3.

Matrix multiplication is more complex. Every element on one row in Matrix1 is multiplied times every element of one column in Matrix2. This means that matrices of different sizes can be multiplied and result in a product matrix that is yet another size.



It may be simpler to look at the completed multiplication of the matrices below. What you see is that Matrix1 (2 X 3) is multiplied times Matrix2 (3 X 2), which results in Matrix3, which is a (2 X 2). This means that the number of rows of Matrix1 must be equal to the number of columns of Matrix2 and the product becomes the number of rwos from Matrix1 times the number of columns from Matrix2.



You are writing a method to multiply two matrices that returns the product to a third matrix.

You are not concerned with checking row and column sizes. Assume that this has been done prior to calling the `multiply` method.

Complete method `multiply` below.

```
/** Precondition:   m1 and m2 are 2D static arrays with unknown int values.
 *                 The number of rows of m1 are equal to the number of cols
 *                 of m2, which allows matrix multiplication.
 *  Postcondition: returns a product matrix that uses the proper mathematical
 *                 rules of matrix multiplication.
 */
public static int[][] multiply(int[][] m1, int[][] m2)
```