**Number of Questions — 4**

**Percent of total test grade — 50**

1. In the space below, you will complete a static method multiplesOf which takes two int parameters, number and count. The method body must **return** an int array containing the first count multiples of number. For example,

* multiplesOf(5, 4) should return the array { 5, 10, 15, 20 }
* multiplesOf(11, 3) should return the array { 11, 22, 33 }
* multiplesOf(1, 15) should return the array   
  { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 }

You **must not** use System.out.print or System.out.println in your method.

Complete the method in the space below:

1. In the space below, you will write a method reverseArray that takes as input an int array and constructs, fills, and returns an array with the order of elements reversed. For example,

* If we declare: int[] arr = { 1, 2, 3, 4, 5 };  
  then reverseArray(arr) should return the array { 5, 4, 3, 2, 1 }
* reverseArray( { 0, -1, 6, -1, 6} ) should return the array { 6, -1, 6, -1, 0 }
* reverseArray( { 0 } ) should return the array { 0 }.

You **must not** use System.out.print or System.out.println in your method.

Complete the method in the space below:

1. In this problem, you will write part of a program to grade students. Suppose we have a String[] names array of student names, and a double[] grades array of student grades. We know that the grades array is the same length as the names array, and the value at grades[index] is the grade for the student named names[index].  
     
   Also suppose that a *passing* grade is 65.0 -- any grade less than this indicates fail, while greater than or equal to 65.0 indicates passing.  
     
   We want to write a method printPassOrFail which prints out all of the students names, and next to each, whether that student passed or failed.  
     
   For instance, if we declare:  
     
    String[] names = { "Peter", "Christine", "Glenn" };  
    double[] grades = { 90.0, 90.0, 50.0 };  
     
   then printPassOrFail(names, grades) prints out:  
     
    Peter: pass  
    Christine: pass  
    Glenn: fail  
     
   If we declare  
     
    String[] names = { "Bill", "Michael", "Rudi" };  
    double[] grades = { 65.0, 64.9, 0.0 };  
     
   then printPassOrFail(names, grades) prints out:  
     
    Bill: pass  
    Michael: fail  
    Rudi: fail  
     
   Your method should not return any values, and you **must** use System.out.print and/or System.out.println in your solution. Write your solution on the following page.

// Prints out student names along with "pass" or "fail" (depending on // whether the corresponding grade is less than or greater than 65).  
// **Precondition**: names.length == grades.length

public static void printPassOrFail(String[] names, double[] grades) {

1. For this problem you will be modifying a multidimensional array.  
     
   Pictures on a computer are usually stored as multidimensional arrays. One way to do this for “black-and-white” images is to break them down into tiny dots called *pixels* and make an two-dimensional int[][] array of these pixels. Each element of the two-dimensional array indicates whether the pixel is dark (value 0), bright (value 255), or somewhere in between (any value 1 to 254). For example:  
     
   int[][] pixels = new int[][]{

{ 255, 255, 255, 255, 255, 255, 255, 255, 255, 255 },

{ 255, 0, 0, 0, 0, 0, 0, 0, 0, 255 },

{ 255, 0, 0, 0, 128, 128, 0, 0, 0, 255 },

{ 255, 0, 0, 0, 128, 128, 0, 0, 0, 255 },

{ 255, 0, 0, 0, 0, 0, 0, 0, 0, 255 },

{ 255, 255, 255, 255, 255, 255, 255, 255, 255, 255 }

};

This array stores an image of a bright rectangle (see the 255 values around the outside) with a dark interior (all those 0 values in the middle) with a gray square in the center (the 128 values). To make an image *brighter*, all we need to do is *increase* (or *add to*) the pixel values of the image.   
  
You will write a method brighten which will take an int[][] pixel array and add a certain amount to every pixel of the array, up to a **maximum of 255**. (If adding amount to an element would cause it to be greater than 255, it must be set to exactly 255).  
  
For instance, with the definition of pixels above, after the call   
 brighten(pixels, 10);  
the array pixels should now contain:

{{ 255, 255, 255, 255, 255, 255, 255, 255, 255, 255 },

{ 255, 10, 10, 10, 10, 10, 10, 10, 10, 255 },

{ 255, 10, 10, 10, 138, 138, 10, 10, 10, 255 },

{ 255, 10, 10, 10, 138, 138, 10, 10, 10, 255 },

{ 255, 10, 10, 10, 10, 10, 10, 10, 10, 255 },

{ 255, 255, 255, 255, 255, 255, 255, 255, 255, 255 }}  
  
Complete the method brighten on the next page.  
  
  
// Brightens the supplied image by adding amount to every  
// element of the pixels array, up to a maximum of 255.  
// **Precondition**: 0 <= pixels[i][j] <= 255 for all i, j  
// **Precondition**: amount >= 0

public static void brighten(int[][] pixels, int amount) {  
  
 int numRows = pixels.length;  
 int numColumns = pixels[0].length;

**END OF SECTION II.**