APCS Exercises Arrays Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

1) Show visually what the array would look like given the following code:

int[] nums = new int[3];

nums[0] = 9;

nums[1] = -4;

nums[2] = nums[1] \* (-1);

2)

int[] nums = new int[4];

nums[3] = 10;

nums[1] = 4;

nums[2] = nums[1] \* 3;

nums[0] = nums[3] + nums[2];

3) Define an array of 100 integers and initialize all values to 25;

4) Given any array, write the code that would find the sum of all odd elements:

int[] nums = {6, 2, 8, 3, 7, 1, 0, 4, 7, 5}; //sum of odd elements would be 23

5) Given any array, write the code that would multiply every value in the array by 10:

6) Show the array that is the result of the following code:

int[] list = new int[10];

for(int i=0; i < list.length; i++)

list[i] = i \* i;

7) Show the array that is the result of the following code:

int[] list = new int[10];

for(int i=0; i < list.length; i++)

{

if(i % 2 == 0)

list[i] = i \* i;

else

list[i] = i + 10;

}

8) Write a method to do the following: when sent a String array as an argument, the method shows all of the elements in their upper-case form.

9) Write a method to do the following: when sent two arguments (a String array called names and a single String called target), the method will count all occurrences of target within the array and return the count.

10) Write a method to do the following: when sent an int array as an argument, the method will return the largest value within the array.

APCS Exercises Array Mechanics Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

1) What would be the output of the following code:

String [] words = new String[3];

String [] monsters = words;

words[2] = “godzilla";

words[1] = “rodan”;

words[0] = “gamera”;

monsters[2] = monsters[2].toUpperCase();

for(int i = 0; i < words.length; i++)

System.out.println(words[i]);

2) If we intend to have two different arrays in #1, one of which has “godzilla” as lowercase and one of which as uppercase, describe what would we do to fix it?

(no code needed here… just a description).

3) Write a method that will accept an argument as an array of String. The method will return a new array of String, with all elements from the argument array copied into it. The returned array must be a different entity from the one sent as an argument, but storing similar values.

4) Rewrite the code in #1, but call your method from #3 so that there are two separate arrays: the second array has a copy of all elements from the first, but the last element is all uppercase.

5) What would be the output of the following code:

int [] nums = new int[10];

for(int i = 0; i < nums.length; i++)

nums[i] = 5;

int index = 0;

while(index < nums.length)

{

nums[index] = nums[index] – index;

index += 2;

}

for(int i=0; i < nums.length; i++)

System.out.print( nums[i] + “ “);

6) Write a method that accepts an array of double as an argument. Assume that each element represents a percentage (out of 100). For each element in the array, change it into its decimal form.

7) Write a method that accepts an array of int as an argument. For each element that has a negative value, change it to zero. Example:

If the array sent as an argument contains: 78 -13 62 68 74 -14 58 -16

The method will change the array to be: 78 0 62 68 74 0 58 0

8) Write a similar method as the one described in #7, but make it so that it returns a new array with the negative values changed to zeroes, and the original array is unaltered.

APCS Exercises 2-DArrays Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

1) Define a 2 dimensional array of doubles with 4 rows and 8 columns.

2) Define an array of ints with 8 rows and 8 columns, and then initialize every element to 4.

3) Create an array that we would visualize like this:

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

4) beets beets

beets beets

beets beets

beets beets

beets beets

beets beets

5) 0 1 2 3 4 5 6

1 2 3 4 5 6 7

2 3 4 5 6 7 8

3 4 5 6 7 8 9

6) 0 0 0 0 0 0 0

0 1 2 3 4 5 6

0 2 4 6 8 10 12

0 3 6 9 12 15 18

7) Write a method to do the following: given a 2-D array of Strings, it asks the user to input values to fill the array for every cell.

Consider the following class definition:

**public** **class** Point

{

**private** **int** x; //data fields

**private int** y;

//constructors

**public** Point() **public** Point(**int** dX, **int** dY)

{ {

x=0; x=dX;

y=0; y=dY;

} }

**public** String toString()

{ **return** “(“ + x + “,” + y + “)”; }

//accessors – get methods

**public int** getX() **public int** getY()

{ **return** x; } { **return** y; }

//mutators – set methods

**public void** setX(**int** dX) **public void** setY(**int** dY)

{ x = dX; } { y = dY; }

}

8) Create a single dimensional array of 35 points, all starting with the value 0,0.

(use the default constructor)

9) Create an array of points with 10 rows and 10 columns starting with 0,0.

10) Create a single dimensional array of points all with the value (3,5).

column index🡪 0 1 2 3

11) Create this array of points: row index🡪 0 (0,0) (0,1) (0,2) (0,3)

1 (1,0) (1,1) (1,2) (1,3)

2 (2,0) (2,1) (2,2) (2,3)

12) Write a method to do the following: given a 2-D array of int, return the sum of all values.

APCS Exercises Regular for-loop vs Enhanced for-loop Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_

1) Write a method that when given an array of String, displays the array elements in reverse:

2) Write a method that will accept an argument as an array of String. The method will return a new array of String, with all elements from the argument array copied into it. The returned array must be a different entity from the one sent as an argument, but storing similar values.

Look familiar? This time, you must use an enhanced for-loop.

3) Write a method to do the following: when sent an int array as an argument, the method will return the largest value within the array. This should also look familiar, but this time use an enhanced for-loop.

4) Write a method to do the following: when sent a 2D array of double, return the sum of all the elements. Again, this time use an enhanced for-loop.