Name _____ Period_____

1. The `DivBySum` class is intended to compute the sum of all the elements in an int array `arr` that are divisible by the int `num`. Consider the following examples,

arr	num	result	Explanation
{4, 1, 3, 6, 2, 9}	3	18	Result is 18 which is the sum of 3, 6, and 9
{4, 1, 3, 6, 2, 9}	5	0	Result is 0 since none of the integers are divisible by 5
{1, 3, 5, 8, 12, 27, 8}	2	28	Result is 28 which is the sum of 8, 12, and 8

Complete the `DivBySum` class <u>using an enhanced for loop</u>. Assume that `arr` and `num` are properly declared and initialized. <u>You must use an enhanced for loop to earn full credit</u>.

```
public class DivBySum{

     public static void main(String args[]){

        int result = 0;
        for(int x : arr){
            if(x % num == 0){
                result += x;
            }
        }









     }
}
```

/3

2. The `WordScrambler` class, takes words from two different arrays, scrambles them, then stores the result in a new array. The words are scrambled by taking the first half of the word from `arr1` and the second half of the word from `arr2` and combining them. Below are examples,

array	word	half	combined
arr1	apple	ap	arap
arr2	pear	ar	

The scrambled words are then stored in a new array called result. Below is an example,

arr1	{“apple”, “bear”, “Timberline”, “Thanksgiving”}
arr2	{“pear”, “light”, “school”, “Friday”}
result	{“arap”, “ghtbe”, “ooltimbe”, “daythanks”}

Complete the `WordScrambler` class below. You may assume that `arr1` and `arr2` are the same size and have been declared and properly initialized.

```
public class WordScrambler{

     public static void main(String args[]){

        String result[] = new String[arr1.length];


        for(int i = 0; i<arr1.length;i++){
            String half1 = arr1[i].substring(0, arr1[i].length()/2);
            String half2 = arr2[i].substring(arr2[i].length()/2);
            result[i] = (half2 + half1).toLowerCase();
        }









   }
}
```

/5

Score _____/

3. The `FindValley` class evaluates whether an array of integers has the *valley* property. An array of positive integers has the *valley* property if the elements are ordered such that successive values decrease until a minimum value (the minimum of the valley) is reached and then the successive values increase.

   The following table illustrates the value assigned to `valleyIndex` for several integer arrays. In each case, if a valley is not found `valleyIndex` has a value of -1, otherwise it has the value of the first valley found.

arr	valleyIndex
{11, 22, 33, 22, 11}	-1
{11, 22, 11, 22, 11}	2
{11, 22, 33, 55, 77}	-1
{99, 33, 55, 77, 120}	1
{99, 33, 25, 77, 55}	2
{33, 22, 11}	-1

Complete the `FindValley` class below. You may assume that `arr` and `valleyIndex` are properly declared and initialized.

```
public class FindValley{

     public static void main(String args[]){


     int valleyIndex = -1;
        for(int i = 1; i < arr.length - 1; i++){
            if(arr[i - 1]>arr[i] && arr[i+1] > arr[i]){
                valleyIndex = i;
                break;
            }
        }









   }
}
```

/5

Score _____/

Score _____/