

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 using an enhanced for loop. Assume that `arr` and `num` are properly declared and initialized. You must use an enhanced for loop to earn full credit.

[illegible]

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", "thanksfri"}

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[]){
```

15

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[]){
```

4. The `GradBookStats` class is intended to compute basic statistics associated with different assignments in a gradebook. Below is a portion of a gradebook.

	Exam 1	Exam 2	Exam 3	Lab 1	Lab 2	Lab 3	Project 1	The number of assignments continues 
Bart	5	1	3	1	2	3	3	
Homer	4	4	4	4	4	4	4	
Wilma	4	5	2	5	3	4	4	
Averages	4.33	3.33	3.0	3.33	3.0	3.66	3.66	

The grade book above can be visualized as a series of parallel arrays as follows,

```
String assignments[] = {"Exam 1", "Exam 2", "Exam 3", "Lab 1", "Lab 2", "Lab 3", "Project 1", ...};
int Bart[] = {5, 1, 3, 1, 2, 3, 3, ...};
int Homer[] = {4, 4, 4, 4, 4, 4, 4, ...};
int Wilma[] = {4, 5, 2, 5, 3, 4, 4, ...};
/* averages array implementation not shown */
```

- (a) In the space below write code that could be used to calculate the average of each assignment, then store the resulting average in a new array called `averages`. Each index in `averages` should map to the appropriate assignment. The number of assignments is not known; therefore, `averages` should be initialized in terms of the length of the `assignments` array. You must also be mindful that the number of digits to the right of the decimal should not exceed 2. For example, the Exam 1 average should be reported as 4.33, not 4.333333333333333

```
public class GradeBookStats{
    public static void main{
```

```
    }
}
```

/5

(b) The mode of a data set refers to the value that occurs most often. The mode for each assignment is shown below.

	Exam 1	Exam 2	Exam 3	Lab 1	Lab 2	Lab 3	Project 1	mode
Bart	5	1	3	1	2	3	3	3
Homer	4	4	4	4	4	4	4	4
Wilma	4	5	2	2	3	3	2	2

In the space below write code that could be used to calculate the mode for a given student. Where `student` represents an array of scores received by a student. For example,

```
int student[] = Bart;
```

The score that occurs most often should be assigned to the variable `mode`.

```
public class GradeBookStats{
    public static void main{
        int Bart[] = {5,1,3,1,2,3,3};
        int Homer[] = {4,4,4,4,4,4,4};
        int Wilma[] = {4,5,2,5,3,4,4};
        int student[] = Bart;
        int mode = 0;
```

```
    }
}
```

/5

5. The `Vocab` class, is used to analyze words in terms of their presence in a vocabulary list. For example,

Consider the vocabulary and word lists below which are stored in the arrays `vocabList` and `wordList`, respectively.

```
String vocabList[] = {"time", "food", "dogs", "cats", "health", "plants", "sports"};
String wordList[] = {"dogs", "toys", "sun", "plants", "time"};
```

The `Vocab` class does the following,

- Counts the number of words in `wordList` that are not in the `vocabList` and stores this value in `countNotInVocab`
- Creates a new array called `missingVocab` that is the same length as the value of `countNotInVocab`
- Stores the missing vocab in the `missingVocab` array

The following example illustrate the behavior of the `Vocab` class.

`vocabList`

"time"	"food"	"dogs"	"cats"	"health"	"plants"	"sports"
--------	--------	--------	--------	----------	----------	----------

`wordList`

"dogs"	"toys"	"sun"	"plants"	"time"
--------	--------	-------	----------	--------

`missingVocab`

"toys"	"sun"
--------	-------

- (a) Write code that could be used to count the number of words in `wordList` that are not found in `vocabList`. The final value should be stored in the variable `countNotInVocab`.

```
public class Vocab{
    public static void main{
String vocabList[] = {"time", "food", "dogs", "cats", "health", "plants", "sports"};
String wordList[] = {"dogs", "toys", "sun", "plants", "time"};
String missingVocab[];
int countNotInVocab = 0;
```

} }

/6