CSCI-101 Exam 3

Name _____

Instructions:

Please follow the rules below as you work through this exam.

- Please leave all notebooks and electronics (including cell phones and smart watches) at the side of the room.
- This is a closed book/closed notes exam.
- **Do not spend too much time on any one problem.** You have 50 minutes to complete this exam.
- Partial credit is awarded.
- Please write legibly. If I cannot read your answers, I cannot give you credit.
- Please write your answers **in the order specified**. If you need additional paper, please raise your hand to ask your instructor for additional paper.
- Your code must be written to behave as specified.
- You must properly use all identifiers that are explicitly stated.
- Please use proper and consistent coding conventions (spacing, naming identifiers, etc.).
- Please stay in your seat until you are ready to hand in your exam. You may leave when you are finished.
- Once you leave the testing room you cannot return until the exam is over. If you need to use the restroom, please use it now.

Instructions for Remote Students

- You are not allowed to use any resources *including the compiler*. You will receive a score of 0 if you are found to be using your compiler.

- Keep your eyes on your computer screen for the duration of the exam. If the instructor suspects you are using resources that are not allowed you will have to retake the exam as an oral exam with the instructor.

- Share your entire screen.

- Open your terminal application and place the terminal application in one half of the screen. Place this document in the other half of the screen.

- In the terminal application, log into cs.bridgewater.edu.

- **In your home directory**, create a directory named **exam3** and change your working directory to **exam3**.

- Create a file named **Exam3.java.** Write a complete program that satisfies the program requirements shown below.

- When you are finished, let me know in the Chat pane. I will then copy your exam files to my computer.

Instructions

Write a class named **Utility** and a class named **Driver**. Provide static methods in the **Utility** class that satisfy the requirements in the section below named Utility. Provide a method named **main** in the **Driver** class that satisfies the requirements in the section below named Driver.

Utility

Define a method named **print** that takes an array of integers as an argument and prints each integer in the array on the same line with spaces between them.

Define a method named **indexOf** that has an array of Strings as the first parameter and a String as a second parameter. If the string passed in as a second argument exists in the array that is passed in as a first argument, then the method returns the index of the first instance of the string found in the array. If the string does not exist in the array, then the method returns -1.

Define a method named **createArray** that has two integer parameters. The method creates and returns an array of integers having a length equal to the value of the first argument and initializes all of the elements in the array so that they are equal to the value of the second argument.

Define a method named **getColumn** that has a 2D array of integers as the first parameter and an integer as the second parameter. The method returns null if the rows of the 2D array have different lengths. The method also returns null if the second argument is not a valid column index. If the lengths of the rows of the 2D array are equal and the second argument is a valid column index, then the method returns a new array containing the values in the column of the 2D array whose index is specified in the second argument.

For example, the method will return an array containing the values {1, 2, 3, 4} if the arguments passed to the method are the 2D array given below and the integer 2.

```
0 0 1 0
0 0 2 0
0 0 3 0
0 0 4 0
```

Driver

Ask the user for 3 strings and add them to an array. Ask the user for another String and store it in a variable. Determine if the 4th string exists in the array by calling the **indexOf** method in the Utility class. If it exists in the array, then print the index of the array for which the first instance of the string is found; otherwise print -1.

Create an array of 20 integers, all initialized to -1, by calling the **createArray** method in the **Utility** class. Print the values in the array by calling the **print** method in the **Utility** class.

Ask the user for 9 integers and populate a 3x3 array with the values entered by the user. Get the array of values in the second column of the array by calling the **getColumn** method in the **Utility** class. Print the values in the array returned by **getColumn** by calling the **print** method in the **Utility** class.