CS 61BL Summer 2026

Introduction to Java Programming

Exam Prep 01: June 26th, 2026

1 Our First Java Program

This exercise is adapted from Head First Java.

Consider the following implementation of a `Dog` class.

```
class Dog {
  public static int cuteness = 100;
  public Dog(String name, int size) {
    // Implementation not provided.
  }

  public void bark(int loudness) {
    // Implementation not provided.
  }

  public static void evolve(int newCuteness) {
    System.out.println("Becoming cuter...");
    cuteness = newCuteness;
  }
}
```

Next to each line below, write out what you think the code will do when run. If you think the line will result in an error, correct it, and proceed through the code as if it is running your corrected version. The `Dog` class is included below for your reference.

```
size = 27; /* Example answer: Errors. size must be declared as an int or other number
type. To fix this error, the line should be "int size = 27;". */
name = "Fido";
Dog myDog = new Dog(name, size);
Dog yourDog = new Dog("Scruffy", 1000);
Dog[] dogList = new Dog[3];
dogList[0] = myDog;
dogList[1] = yourDog;
dogList[2] = 5;
dogList[3] = new Dog("Cutie", 8);
int x;
x = size - 5;
if (x < 15) {
  myDog.bark(8);
}
myDog.evolve(200);
Dog.evolve(300);
Dog.bark(10);
```

2 Mystery

This is a function (a.k.a. method). It takes an array of integers and an integer as arguments, and returns an integer.

```
public static int mystery(int[] inputArray, int k) {
  int x = inputArray[k];
  int answer = k;
  int index = k + 1;

  while (index < inputArray.length) {
    if (inputArray[index] < x) {
      x = inputArray[index];
      answer = index;
    }
    index = index + 1;
  }

  return answer;
}
```

(a) What would `mystery` return if `inputArray = [3, 0, 4, 6, 3]` and `k = 2`?

(b) Explain, in natural language, what `mystery` does.

(c) Use `mystery` to sort a given array of integers in ascending order. *Hint: Can you use what mystery returns to swap integers within the array?*

```
public static void sortArray(int[] inputArray) {
```

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```

3 Two Sum

Given an array of numbers **`nums`**, find two numbers in **`nums`** such that adding them would result in the value **`target`**. Assume that only one such pair exists, and there is always a pair that fulfills the condition.

Return the numbers as an array with two terms.

You may not need all lines provided.

```
int[] twoSum(int[] nums, int target) {
```

```
  return new int[]{-1, -1}; // No solution, technically wouldn't happen.
}
```

4 Reversals

In this problem, we are trying to implement a **`reverseString`** method. This method takes in a string and reverses it.

```
public static String reverseString(String s) {
  // Implementation to be filled in
}
```

(a) For each of the test cases we have written below, write down whether it is correct, and briefly explain why it is, or isn't useful.

Some questions to guide your thinking: Does the test case actually test the method **`reverseString`**? Does it test the method with a variety of inputs? Does it test edge cases?

```
assertThat("hello world").isEqualTo("dlrow olleh");
```

```
assertThat(reverseString("Hello")).isEqualTo("olleH");
```

```
assertThat(reverseString(1234)).isEqualTo(4321);
```

```
assertThat(reverseString("@#_!")).isEqualTo("!_#@");
```

```
assertThat(reverseString("")).isEqualTo("");
```

```
assertThat(reverseString("baaaa")).isEqualTo("aaaab");
```

(b) Consider the following implementation for **reverseString**

```
public static String reverseString(String s) {
  return s.substring(1) + s.charAt(0);
}
```

Here is an explanation of some features of the code, as well as an example: **substring(i)** returns a substring that begins with the character at the specified index and extends to the end of this string. **charAt(i)** returns the character at index **i**. Note that 0-indexing also applies to strings. The + sign concatenates the two things returned by the two function calls into a longer string.

If the index is out of bounds, then an **IndexOutOfBoundsException** is thrown, and the program errors.

```
s = "Circle";
s.substring(2); // Returns "rcle"
s.charAt(1); // Returns 'i'
s.substring(2) + s.charAt(1) // Returns "rclei"
s.substring(6) // Errors!
```

Some of the test cases from the previous part are replicated below. Write down what our method would return (could be an error), and determine whether each of these would pass for our implementation.

```
assertThat(reverseString("Hello")).isEqualTo("olleH");
```

```
assertThat(reverseString("")).isEqualTo("");
```

```
assertThat(reverseString("baaaa")).isEqualTo("aaaab");
```

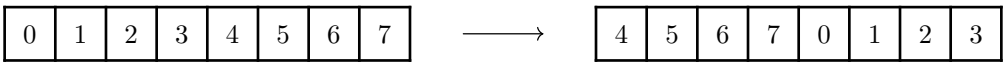
(c) Write a correct implementation for **reverseString**. You may not need all lines provided.

Hint: use **charAt(i)**, which takes in an integer and returns the character at that index. Remember that strings are 0-indexed, so the last character of a string **s** is **s.charAt(s.length() - 1)**.

```
public static String reverseString(String str) {
  _____________________________________________________________
  _____________________________________________________________
  _____________________________________________________________
  _____________________________________________________________
  _____________________________________________________________
  _____________________________________________________________
  return reversed;
}
```

5 Rotate

Write a function that, when given an array **A** and integer **k**, returns a new array whose contents have been shifted **k** positions to the right, wrapping back around to index 0 if necessary. For example, if **A** contains the values 0 through 7 inclusive and **k = 12**, then the array returned after calling **rotate(A, k)** is shown below on the right:



k can be arbitrarily large or small - that is, **k** can be a positive or negative number. If **k** is negative, shift **k** positions to the left. After calling **rotate**, **A** should remain unchanged. (This means that **rotate** is **nondestructive!**)

Hint: you may find the modulo operator % useful. Note that the modulo of a negative number is still negative (i.e. (-11) % 8 = −3).

```
/** Returns a new array containing the elements of A shifted k positions to the right. */

public static int[] rotate(int[] A, int k) {

    int rightShift = _____;

    if (_____) {

        _____;
    }


    int[] newArr = _____;

    for (_____) {

        int newIndex = _____;

        _____;
    }
    return newArr;
}
```