University of Maryland College Park
Department of Computer Science
CMSC131 Fall 2025
Exam #1

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE): KEY

STUDENT ID (e.g. 123456789):

Instructions

- Please print your answers and use a pencil.
- Do not remove the staple from the exam. Removing it will interfere with the Gradescope scanning process.
- To make sure Gradescope can recognize your exam, print your name, write your directory id at the bottom of pages with the text DirectoryId, **provide answers in the rectangular areas provided**, and do not remove any exam pages. Even if you use the provided extra pages for scratch work, they must be returned with the rest of the exam.
- This exam is a closed-book, closed-notes exam, with a duration of 50 minutes and 100 total points.
- Your code must be efficient.
- Multiple choice questions have only one answer unless indicated otherwise.
- You don't need to use meaningful variable names; however, we expect good indentation.

Grader Use Only

#1	Problem #1 (Short Answers – 2 pts each)	16
#2	Problem #2 (Code 1)	22
#3	Problem #3 (Code 2)	22
#4	Problem #4 (Code 3)	40
Total	Total	100

Problem #1 (Short Answers – 2 pts)

1. Which statement is **false**?
   a. The expression 10 / 3 in Java evaluates to 3.
   b. Calling **`nextLine`**() after **`nextInt`**() may appear to "skip" input because the newline is still in the buffer.
   c. The keyword **boolean** can store either **`true`** or **`false`**.
   d. The keyword **`float`** in Java has more precision than **`double`**.

2. Which statement is **true**?
   a. In Java, the **==** operator compares the contents of two **`String`** objects.
   b. A **`for`** loop must always execute at least once.
   c. 8_{10} in octal would be 10_8
   d. You can declare a variable with the name: **`class`**.

3. The component of the Java platform responsible for executing compiled bytecode and providing platform independence is called the **Java Virtual Machine (or JVM)**.

4. The region of a program where a variable can be accessed or modified is called its **scope**.

5. 87_{10} is what in binary (answer must have 8 bits) (no calculators) ? **01010111 (must have leading 0)**

6. Is this a valid variable declaration: `int 2numbers;` Either answer valid or explain why it is not valid.

Variable name cannot start with a digit

7. Assume the following code fragment in a **`main`** method, what is the **output of the very last line**? **7**

```
for (int i = 1; i <= 4; i++) {

    for (int j = 1; j <= 2; j++) {

        System.out.println(i + j);
        ++i;
    }
}
```

8. Assume the following code fragment in a **`main`** method, what is the output? **10**

```
int a = 4;
int b = 2;
int c = (a++) + (++b);
if (c % 2 == 1 && a + b == 7)
    System.out.println(c + a);
else if (b-- > 2 || ++a == 6)
    System.out.println(a * b);
else
    System.out.println(c - b);
```

Problem #2 (Code 1)

Write a public static method called `replaceAtIndex` which returns a `String` and has three parameters: a `String` called `text` (assume a String with at least one character), an `int` called `index`, and a `char` called `newChar`. The method should return a new string that is the same as the original string except that the character at position `index` is replaced by `newChar`. If the index is less than 0 or greater than or equal to the length of the string, the method should simply return the original string without any changes. As for library methods, you may only use the `length()` and `charAt()` methods from the String class.

```
public class Code1 {
    public static String replaceAtIndex(String text, int index, char newChar) {

      if (index < 0 || index >= text.length()) {
            return text; // invalid index
        }

        String result = "";
        for (int i = 0; i < text.length(); i++) {
            if (i == index) {
                result += newChar;  // replace at index
            } else {
                result += text.charAt(i); // copy original char
            }
        }
        return result;
    }
```

```
      public static void main(String[] args) {
      System.out.println(replaceAtIndex("hello", -1, 'x'));  // hello
      System.out.println(replaceAtIndex("world", 5, '!'));   // world
      System.out.println(replaceAtIndex("hello", 1, 'a'));   // hallo
      System.out.println(replaceAtIndex("world", 4, '!'));   // worl!
      System.out.println(replaceAtIndex("Java", 0, 'j'));    // java
    }
}
```

Directory id:

Problem #3 (Code 2)

Write a public static method called `mysteryTriangle` which returns a `String` and has one `int` parameter called `n`. The method should return a string that represents a triangle of numbers. The first row contains a single number (1), and each subsequent row contains one more number than the previous row. Each number in the triangle is calculated based on the previous one using a simple arithmetic relationship (which you need to figure out), starting with 1 as the first number. Numbers within a row should be separated by a space. If `n` is less than 1, the method should return an empty string. You may not use any library methods. No need for a **`main`**.

Sample call	Output of the call
`System.out.println(mysteryTriangle(5));`	1 3 7 15 31 63 127 255 511 1023 2047 4095 8191 16383 32767

```
public static String mysteryTriangle(int n) {

        if (n < 1) {
            return ""; // return empty string if n is less than 1
        }

        int num = 1; // first number in triangle
        String result = "";

        for (int row = 1; row <= n; row++) {
            for (int col = 1; col <= row; col++) {
                result += num + " ";
                num = num * 2 + 1; // new number logic
            }
            result += "\n";
        }

        return result;
    }
```

Problem #4 (Code 3)

Write two static methods: a helper method named `min` and another method named `numberTool`. The `main` method has been provided for you. First, write the **private static method** called `min` that takes two integer parameters and returns the smaller of the two numbers. If the two numbers are the same, just return the number. Use your `min` method as needed in `numberTool`.

After writing `min`, write a public static method called `numberTool` which returns a `String` and has three integer parameters named `choice`, `first`, and `second`. If either `first` or `second` is less than 1, just return the string `"invalid input"`.

If `choice` is 1, determine the correct ascending order of the three numbers: `first`, `second`, and `131`. Return a single string with the three numbers in sorted order, separated by spaces.

If `choice` is 2, find the smaller of `first` and `second` and calculate the sum of all even numbers from 2 up to and including that **smaller number**. Return the sum as a string.

For any other value of `choice`, return `"invalid choice"`. No library methods allowed in your code.

```
public class ExamProblem {
   private static int min(int a, int b) {

    if (a <= b) {
       return a;
    } else {
       return b;
    }
  }
```

```
public static String numberTool(int choice, int first, int second) {
```


Directory id:

```
        if (first < 1 || second < 1) {
            return "invalid input";
        }

        if (choice == 1) {
            // Sort first, second, 131 using helper min
            int smallest, middle, largest;

            // Find smallest
            smallest = min(min(first, second), 131);

            // Find largest
            if (first >= second && first >= 131) {
                largest = first;
            } else if (second >= first && second >= 131) {
                largest = second;
            } else {
                largest = 131;
            }

            // Find middle: it's the remaining one
            middle = first + second + 131 - smallest - largest;

            return smallest + " " + middle + " " + largest;

        } else if (choice == 2) {
            // Sum even numbers up to smaller of first and second
            int limit = min(first, second);
            int sum = 0;
            for (int i = 2; i <= limit; i += 2) {
                sum += i;
            }
            return "" + sum;

        } else {
            return "invalid choice";
        }
    }
```

```
public static void main(String[] args) {
        System.out.println(ExamProblem.numberTool(1, 135, 67));  //67 131 135
        System.out.println(ExamProblem.numberTool(1, 205, 200)); //131 200 205
        System.out.println(ExamProblem.numberTool(1, 131, 131)); //131 131 131
        System.out.println(ExamProblem.numberTool(1, -5, 10));  //invalid input
        System.out.println(ExamProblem.numberTool(2, 12, 8)); //20
        System.out.println( ExamProblem.numberTool(2, 7, 3)); //2
        System.out.println(ExamProblem.numberTool(3, 5, 10)); //invalid choice. }}
```