



# University of Maryland College Park

## Department of Computer Science

### CMSC131 Fall 2024

### 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 0)	15
#3	Problem #3 (Code 1)	17
#4	Problem #4 (Code 2)	24
#5	Problem #5 (Code 3)	28
<b>Total</b>	Total	100

## Problem #1 (Short Answers – 2 pts)

1. (2 pts) Which statement is false?
  - a. A literal such as 3 would be considered an **int** type in Java.
  - b. **A literal such as 3.14 would be considered a float type in Java.**
  - c. **String** is not a Java primitive type.
  - d. **34 % 34** is 0
2. (2 pts) Which statement is true?
  - a. You can only comment in Java using //
  - b. A method can only have one **return** statement in it.
  - c. **The = operator associates from right to left.**
  - d. The name **\$\_** is not a valid Java variable name.
3. (2 pts) When you compile java source code you get this type of platform independent code? **bytecode**
4. (2 pts) Show how you would create a named constant of type **int** called **NUM** and assign to it **35** (all in one statement).

**final int NUM = 35;**

5. (2 pts) The base 10 number 287 is what in hexadecimal (base 16) (no calculators) ?

**11F**

6. (2 pts) Assume the following code fragment in a **main** method, what is the output of the very last line?

```
for (int i = 0; i < 100; i++)
{
    for (int j = 0; j <= 100; j++ )
    {
        System.out.println(i*j);
    }
}
```

**9900**

7. (2 pts) Assume the following code fragment in a **main** method, what is the output? **12**

```
int x = 5;
int y = x++;

if (y < 5 && ++x == 7)
    System.out.println(x);
else if (y == 6)
    System.out.println(--x);
else
    System.out.println(x*=2);
```

8. (2 pts) Assume the following code fragment in a **main** method, what is the output? **51 51 56 56**

```
for(int i= 1; i <= 10; i+=5)    {
    for(int j=1; j<= 2; j++){
        System.out.print("5" + i + " ");
    }
}
```

## **Problem #2 (Code 0)**

Write a public static method called **StringDemo** which returns the string literal **Same** if both of its arguments have the same length, otherwise it returns the argument with the longer length. You can assume non-nulls String will be passed in as arguments. As for library methods, you can only use the String **length** method. If you have a **System.out.println** in your code, you are not writing the code correctly. Do not change the given **main**.

```
public class Code0 {

    public static String StringDemo(String s1, String s2) {

        if (s1.length()==s2.length())
            return "Same";
        else if (s1.length()>s2.length())
            return s1;
        else
            return s2;
    }

}
```

```
public static void main(String[] args) {
    System.out.println(StringDemo("Java", "else")); //Same
    System.out.println(StringDemo("Java", "if")); //Java
    System.out.println(StringDemo("Java", "while")); //while
}
```

```
}
```

### **Problem #3 (Code 1)**

Write a public static method called **printUpsideDownTriangle** which displays an upside-down triangle with number of rows being equal to its parameter **rows**. Assume that **rows** will be 2 or larger. For this method, you can only use **one loop** (no nested loops). As for library methods, you can only use the **repeat** String methods and **System.out.println**. No need for a **main**.

Sample call	Output of the call
<code>printUpsideDownTriangle(5);</code>	***** **** *** ** *

```
public static void printUpsideDownTriangle(int rows) {  
  
    for (int i = rows; i >= 1; i--) {  
        // Print the asterisks for the current row  
        String stars = "*".repeat(i);  
        System.out.println(stars);  
    }  
}
```

### **Problem #4 (Code 2)**

Write a public static method called **makeString** which returns a **String** that is the concatenation of every character immediately to the right of each instance of character parameter **c** in the string **s**, unless that character is itself **c**. If there is an occurrence of **c** as the last character in the String parameter **s**, there will be nothing to the right to concatenate. Assume a non-null String, and if **s** has no occurrence of **c**, simply return the empty string. As for library methods, you can only use the String **length** and **charAt** method. If you have a **System.out.println** in your code, you are not writing the code correctly. Do not change the given **main**.

```
public class Code2 {

    public static String makeString(String s, char c) {

        String temp = "";
        for (int i = 0; i < s.length()-1; i++) {
            if(s.charAt(i) == c && s.charAt(i+1) != c)
                temp+=s.charAt(i+1);

        }
        return temp;
    }

}

public static void main(String[] args) {
    System.out.println(makeString("abacaaa", 'a')); //bc
    System.out.println(makeString("abacaaa", 'b')); //a
    System.out.println(makeString("abacaaa", 'd')); // empty string

}
```

## Problem #5 (Code 3)

Write a public static method called **printStaircase** which returns a **String** in the shape of a staircase with the number of steps being the parameter **steps**. A step is **\*\*\*\*** on its own line with 2 \* each on its own line under the 4<sup>th</sup> \* of the first row. Each new steps begins one column over from the previous step. Assume **steps** will be 2 or more. No library methods allowed, but you can invoke the given **spaces** method below as you see fit. If you have a **System.out.println** in your code, you are not writing the code correctly. In the code you write, you can only have one loop and it must be a while loop.

Sample call	Output of the call
<code>System.out.println( printStaircase(3) );</code>	<pre>****  *  *   ****    *     *      ****       *        *</pre>

```
public class Stairs {

    public static String spaces(int num) {
        String s = "";
        for (int j=1 ; j<=num; j++)
        {
            s+=" ";
        }
        return s;
    }

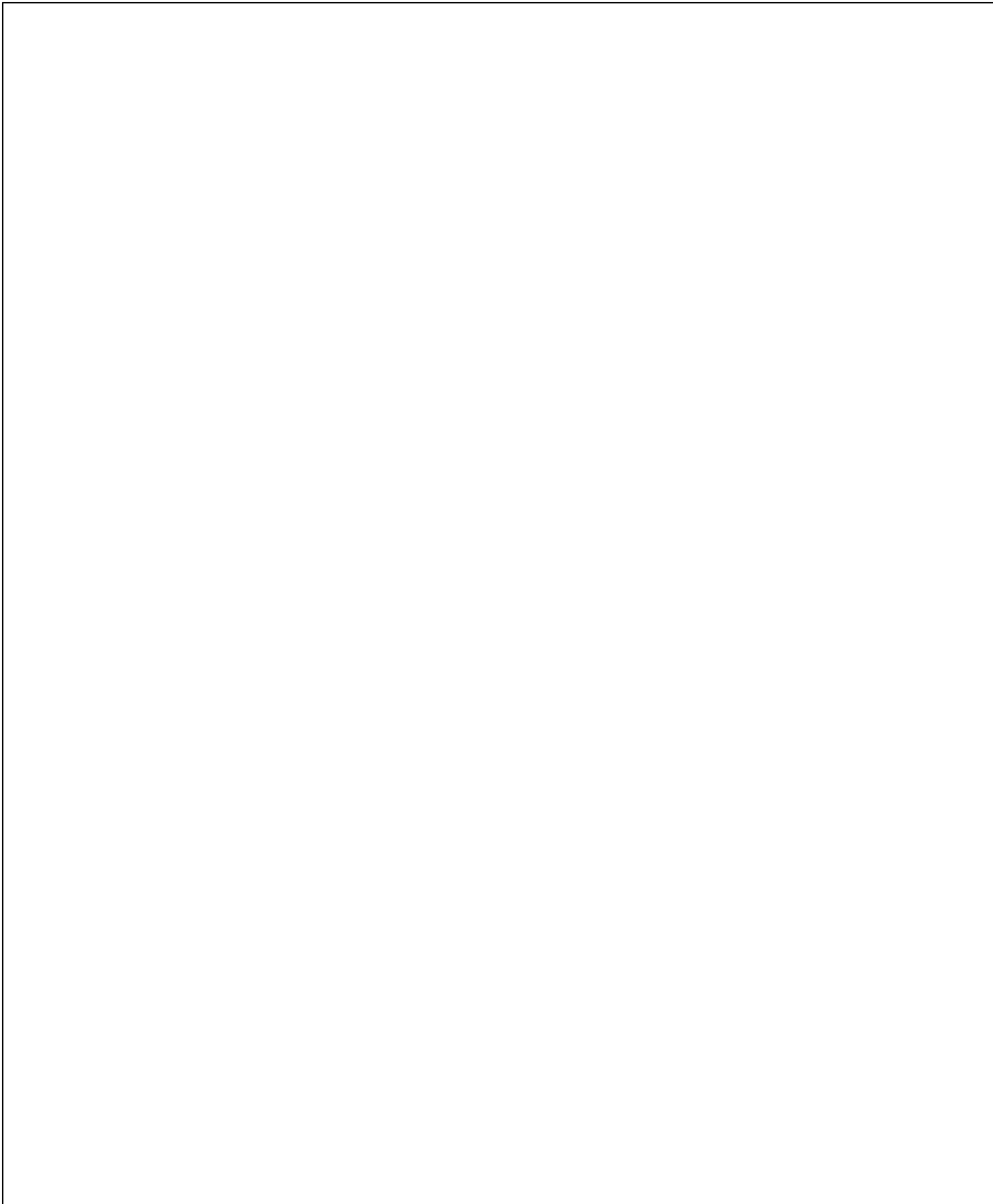
    public static String printStaircase(int steps) {

        int count = 0;
        int i=1;
        String s="";
        while (i<=steps)
        {

            s+= spaces(count) ;
            s+="****\n";
            count+=4;
            s+= spaces(count -1) ;
            s+="*\n";
            s+= spaces(count -1) ;
            s+="*\n";
            i++;
        }

        return s;
    }

}
```



**Extra Page If You Need It**

**Last Page**