CSCI 2110 Data Structures and Algorithms
Fall 2024
Laboratory No. 6
Week of October 21 – October 25

Due (on Brightspace): Sunday, October 27, 11.59 PM

RECURSION

The objective of this lab is to help you to continue to get familiar with recursion by writing several small programs with recursive methods. Your task is to write, compile and run each program using an Integrated Development Environment (IDE). You may use any IDE of your choice – IntelliJ, Eclipse, NetBeans, VSCode, etc.

Remember the rule in recursion: more thinking, less coding!

Marking Scheme: Throughout this course, we will be focusing on the complexity of algorithms and consequently, the efficiency of programs. In your first-year courses, the focus was on creating runnable code to specifications. This course builds on previous knowledge, but also focuses on efficiency. This will involve reducing unnecessary coding and designing or choosing good algorithms.

Each exercise will list the test cases that you need to test the code. Ensure that the output that you generate covers all the test cases.
Each exercise carries 10 points.
Your final score will be scaled down to a value out of 10. For example, if there are three exercises and you score 9/10, 10/10 and 8/10 on the three exercises, your total is 27/30 or 9/10.

Commenting your code: Your code should be clear and easy to understand. While extensive comments are not required, you are expected to include brief, meaningful comments to explain complex logic or any non-obvious sections of your code. Use inline comments to clarify key parts of the code. For methods and classes, use Javadoc to briefly describe their purpose, parameters, and return values. This ensures others can understand your code's intent quickly.

Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.

Submission: One zip file containing the source codes (.java files) for all classes, and sample outputs. Please see details at the end of this document.

Submission Deadline: Sunday, October 27, 2024, 11.59 PM

Grace Time: Submissions will be accepted until 4.59 AM on Monday, October 28, 2024 without late penalty.

Late Penalty: Submissions received after the grace time will be subject to a 10% per day late penalty. The submission portal will close on Wednesday, October 30, 4.59 AM.

Dropping of Labs: Up to 2 labs can be dropped during the semester. No SDA submission is required.

Exercise 1 (Recursive Review)

This exercise presents a brief review of recursion. You will write a series of short methods to implement the recursive programs discussed in the lectures. Each program is presented in pseudo code. Implement all 3 methods as static methods and test them in the main method in a file called Exercise1.java.

a) Factorial of an integer n:

```
if n == 0, then factorial(n) = 1 //base case
if n > 0, then factorial(n) = n * factorial(n-1) //glue case
```

Write a recursive method that calculates the factorial of a non-negative integer *n*. Call this method in your main method, prompting a user to enter a positive integer n and find the factorial of that number. Your method header should be:
public static int factorial(int n)

b) Fibonacci series:

```
if n==0, then fib(n) = 0 //base case
if n==1, then fib(n) = 1 //base case
if n>1, then fib(n) = fib(n-1) + fib(n-2) //glue case
```

Write a recursive method that finds the *n*th number in the Fibonacci series. Call this method in your main method, using a loop to print the first 20 numbers in the Fibonacci series. Your method header should be:
public static int fib(int n)

c) Exponentiation (*x* to the power *n*):

```
if n == 0, then power(x,n) = 1 //base case
if n>0, then power(x,n) = power(x,n-1)*x //glue case
```

Write a recursive method that calculates *x* to the power *n*, where *x* and *n* are both positive integers. Call this method in your main method, **prompting a user to enter *x* and *n***, and print *x* to the power of *n*. Your method header should be:
public static int power(int x, int n)

Test your program to ensure its proper operation. Save data from one test run in a text file.

A sample test run might look like this:

```
Factorial of a number
Enter a positive integer: 4
The factorial of 4 is 24

Fibonacci numbers
The first 20 numbers in the Fibonacci series are
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181

Power of a number
Enter a positive integer x: 2
Enter another positive integer: 3
2 to the power of 3 is 8
```

Exercise 2: Write a recursive method called *countDown* that takes a single positive integer *n* as parameter and prints the numbers *n* through 1 followed by 'BlastOff!'. Your method header should be:
public static void countDown(int n)

countDown(10) would print:

```
10      9       8       7       6       5       4       3       2       1       BlastOff!
```

Write a program called Exercise2.java. Your program should **accept input from a user** (specifying an integer *n*) and count down to 1 from *n* before printing 'BlastOff!'. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.

Exercise3 (Countdown Part 2)

Modify the *countDown* method you wrote in the previous exercise so that it prints only even numbers when *n* is even, and only odd numbers when *n* is odd. Your main method and the header for your *countDown* method can remain unchanged.

countDown(10) would print:

```
10      8       6       4       2       BlastOff!
```

countDown(9) would print:

```
9       7       5       3       1       BlastOff!
```

Write a program called Exercise3.java. Your program should **accept input from a user** (specifying an integer *n*) and count down from *n* before printing 'BlastOff!'. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.

Exercise4 (Multiples)

Write a recursive method called *multiples* to print the first *m* multiples of a positive integer *n*. Your method header should be:
public static void multiples(int n, int m)

multiples(2, 5) would print:
2, 4, 6, 8, 10

multiples(3, 6) would print:
3, 6, 9, 12, 15, 18

Write a program called Exercise4.java. Your program should **accept input from a user** (specifying integers *n* and *m*) and print multiples. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.
Note: You may print multiples in either ascending or descending order.

Exercise5 (Write Vertical)

Write a recursive method called *writeVertical* that takes that takes a single positive integer *n* as a parameter and prints the digits of that integer vertically, one per line. Your method header should be:
public static void writeVertical(int n)

writeVertical(1234) would print:

```
1
2
3
4
```

Write a program called Exercise5.java. Your program should **accept input from a user** (specifying an integer *n*) and print its digits vertically. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.

Exercise6 (Sum of Squares)

Write a recursive method called *squares* that takes that takes a single positive integer *n* as a parameter and calculates the sum of the squares of all digits 1 through *n*. Your method header should be:

```
public static int squares(int n)
```

squares(4) would return:
30

Write a program called Exercise6.java. Your program should **accept input from a user** (specifying an integer *n*) and return the sum of the squares of all digits 1 through *n*. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.

Exercise7 (Reverse a String recursively)

Write a recursive method that takes a string and returns the string in reverse order. Your method header should be:

```
public static String reverseString(String str)
```

For example:

- `reverseString("hello")` returns `"olleh"`.
- `reverseString("recursion")` returns `"noisrucer"`.

Write a program called Exercise7.java. Your program should **accept input from a user** (a String) and display the reversed String. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.

Exercise8 (Count Ways to Climb Stairs)

Imagine you are climbing a staircase. You can take 1 step, 2 steps, or 3 steps at a time. Write a recursive method to calculate how many distinct ways there are to climb to the top of a staircase with n steps.
For example, if you have a staircase with 4 steps, there are 7 ways to climb it:
{1,1,1,1}, {1,1,2}, {1,2,1}, {2,1,1}, {2,2}, {1,3}, {3,1}

Your method header should be:

```
public static int countWays(int n)
```

For example:

- Input: n = 4
- Output: 7

As another example,

- Input: n = 9
- Output: 149

Hint: For the base cases, If n == 0, there's 1 way to stay at the bottom. If n < 0, there's no valid way to climb negative steps. (That is, if n ==0, return 1; if n<0, return 0).

Write a program called Exercise8.java. Your program should **accept input from a user** (a String) and display the reversed String. Test your program for three different inputs to ensure its proper operation. Do not hard code inputs. Save data from the three test runs in a text file.

What to submit:
One zip file containing the following:
1. All .java files
2. One text/word/pdf document containing sample inputs/outputs. Please ensure your name and Banner ID is on this document.

You MUST SUBMIT .java files that are executable by your TAs. TAs will run your program on the RandomNames.txt file and check your solution. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers.