Recursion Competency #7

Recursion	Mastered 1 point	Progressing 0.001 points	Novice 0 points	Criterion Score
Can describe what a recursive method is and the benefits of using recursion	✓			1 / 1
Can develop recursive methods for recursive mathematical functions	✓			1 / 1
Can explain how recursive method calls are handled in a call stack				/ 1
Can solve problems using recursion	✓			1 / 1
Can use an overloaded helper method to derive a recursive method				/ 1
Can implement selection sort using recursion	✓			1 / 1
Can implement binary search using recursion	✓			1 / 1
Can explain the difference between tail recursion and head recursion				/ 1

Explain how recursive method calls are handled in a call stack:

- When the program calls the function, the function goes to the top of the stack data structure. Similar to dishes, stacking the dish on a pile of dishes will force the next person to grab the dish on the top. The program continues to work the program, or work towards the bottom of the dishes. If the condition is true, it floats back to top of the call stack. The program executes the functions until it hits the “bottom plate”, the recursion base. Then when the condition becomes true, the result is sent to the computer.

Explain the difference between tail recursion and head recursion

- Head recursion: when a recursive function calls itself and its the first statement in the function. The processing occurs after the recursive call, similar to like the top of the to-do-list.
- Tail recursion: when a recursive function calls itself and its the last statement in the function.The processing occurs before the recursive call. If the recursive function doesn’t execute, the operation terminates and returns nothing.

*RecursiveMethodsDriver.java

```java
package tryIt;

import java.util.Scanner;

public class RecursiveMethodsDriver {
    public static int fibonacci(int n) {
        if (n == 1) {
            return 1;
        } else if (n == 2) {
            return 1;
        } else {
            return fibonacci(n - 1) + fibonacci(n - 2);
        }

    }

    public static void m1(int n) {
        System.out.println(n);

        if (n > 1) {
            m1(n - 1);
        }
    }

    public static void m2(int n) {
        if (n > 1) {
            m2(n - 1);
        }
        System.out.println(n);
    }

    public static void m3(int n) {
        System.out.println(n);
        m1(n);
        m2(n);
        System.out.println(n);
    }

    public static void m4(int n) {
        for (int i = 0; i < n; i++)
            System.out.print("*");
        {
            System.out.println("");
        }
        if (n > 1)
            m4(n - 1);
        for (int j = 0; j < n; j++)
            System.out.print("!");
        {
            System.out.println("");
        }
    }

    public static void main(String[] args) {
        Scanner userInput = new Scanner(System.in);
        System.out.println("Please enter a number: ");
        int num = userInput.nextInt();
        userInput.nextLine();
        System.out.println(fibonacci(num));
        userInput.close();

        m1(5);
        System.out.println("");
        m2(5);
        m3(5);
        m4(5);

    }
```

Console

<terminated> RecursiveMethodsDriver [Java Application]

```
Please enter a number:
5
5
5
4
3
2
1

1
2
3
4
5
5
4
3
2
1
1
2
3
4
5
*****
****
***
**
*
!
!!
!!!
!!!!
!!!!!
```

Part 1. Tracing Recursive methods

WITHOUT using the computer, find the output of the following method when n = 4.

```
public static int mystery(int n){
      if (n == 0)
            return 0;
      else
            return n*n + mystery(n-1);

}
```

30

Part 2. Implementing Recursive Fibonacci method

In this part of the try-it, you will practice writing a recursive method:

1. Create a java class, called `RecursiveMethodsDriver`, that includes a `main` method.
2. In `RecursiveMethodsDriver`, write a static recursive method, called `fibonacci`, that implements the following recurrence relation:

```
fib( 1 ) = 1          (base case)
fib( 2 ) = 1          (base case)
fib( n ) = fib( n-1 ) + fib( n-2 )
```

3. Add statements in `main` to ask the user to enter a number, then call `fibonacci` method with the number as input and display the output of the method on the screen.
4. What is the value returned with the input number is 10? 55

5. What is the value returned when the input number is 40? 102334155
6. What is the value returned when the input number is 47? -1323752223

7. What is the output when n = -1? Threw an exception
8. What happened in step 7 above? Returns stack overflow error

3. Add a statement in main to call the method `m2` with input 5.

4. Why are the outputs of `m1` and `m2` different?

Because m1 method is printing the n first, then n value after the first method call. While m2 method implements the method first then prints out the n value.

5. Write a similar recursive method called `m3` which displays the following output when given the input 5. Write a statement in main to call `m3` and make sure it gives the correct output.

```
5
4
3
2
1
1
2
3
4
5
```

6. Write another recursive method called m4 which displays the following output when given the input 5. Write a statement in main to call `m4` and make sure it gives the correct output. Hint: m4 is similar to m3 except that instead of printing the number, n, use a for loop to print `n` asterisks or n exclamation points.

```
*****
****
***
**
*
!
!!
!!!
!!!!
!!!!!
```