Recursive Function

A breif recap

- Recursive functions in Java call themselves and are used to solve problems that can be broken down into smaller subproblems.
- They have a base case that eventually stops the recursion.
- Examples of problems that can be solved with recursive functions include factorials, Fibonacci sequences, and binary search.

Critical Thinking

- What is the purpose of the following recursive function?

```
public static int mystery(int a, int b) {
    if (b == 0) {
        return a;
    } else {
        return mystery(a, b -1) + 1;
    }
}
```

answer: This function returns the sum of a and b.

- I want this function to calculate the multiplication of positive integers a and b. What should be in blank space?

```
public static int multiplication(int a, int b) {
    if (a == 1) {
        ____________;
    } else {
        return multiplication(a, b -1) + b;
    }
}
```

answer: return b;

- what will happen if I call a()?

```
public static void a() {
    b();
}
public static void b() {
    a();
}
```

answer: The program will crash because the function will call itself infinitely. (it crash because system run out memory)

- What will happen when we call a(n) with a non-negative n value?

```
static void a(int n) {
        if(n==0)
            System.out.println("ends in a()");
        else
            b(n-1);
    }
static void b(int n) {
        if(n==0)
            System.out.println("ends in b()");
        else
            a(n-1);
}
```

answer: It ends in a() if n is even, and ends in b() if n is odd.

Let's say n=5

a(5) –> b(4) –> a(3) –> b(2) –> a(1) –> b(0) prints "ends in b()"

- In previous question, what will happen when we call b(n) with a non-negative n value?

answer: infinite recursion

Practice

- Write a recursive function that takes two non-negative integers "n" and "m" and computes the power n^m. you are not allowed to use "for" or "while".

  example:

  - power(2, 3) –> 8

soloution1:

```
public static int pow(int a, int b) {
    if(b==1)
        return a;
    else
        return pow(a, b-1) * a;
}
```

soloution2:

```
// not tested
public static int power(int n, int m) {
    if (m == 0) {
        return 1;
    } else {
        return n * power(n, m - 1);
    }
}
```

- Write a recursive function which takes two positive integer arguments n,m and returns n % m. Don't use %, *, / operators.

  example:

  - mod(3, 2) –> 1

soloution:

```
public static int mod(int a, int b) {
    if(a<b)
        return a;
    else
        return mod(a-b, b);
}
```

- Write a recursive function which takes a string parameter and checks if all its characters appear only once.

  example:

  - isUnique("pickle") –> true
  - isUnique("moon") –> false
  - isUnique("trash") –> true

soloution1:

```
// not tested
public static boolean isUnique(String s) {
    if(s.length() == 1)
        return true;
    else if(s.charAt(0) == s.charAt(s.length()-1))
        return false;
    else
        return isUnique(s.substring(1, s.length()-1));
}
```

soloution2:

```
public static boolean isUnique(String s) {
    if(s.length() == 1)
        return true;
    else {
        for(int i=1; i<s.length(); i++)
            if(s.charAt(0) == s.charAt(i))
                return false;
        return isUnique(s.substring(1));
    }
}
```

- Write a method that takes three integer arguments and returns their maximum. (You can use Math.max() function)

soloution:

```
public static int maxThree(int a, int b, int c) {
    return Math.max(a, Math.max(b, c));
}
```

Project

// TODO

1. Write a recursive function to calculate the factorial of a number.
2. Write a recursive function to find the nth number in the Fibonacci sequence.
3. Write a recursive function to calculate the sum of an array of integers.
4. Write a recursive function to reverse a string.
5. Write a recursive function to find the maximum value in an array of integers.
6. Write a recursive function to check if a given string is a palindrome.
7. Write a recursive function to count the number of occurrences of a given character in a string.
8. Write a recursive function to find the greatest common divisor (GCD) of two numbers.
9. Write a recursive function to check if a given binary tree is a binary search tree (BST).
10. Write a recursive function to merge two sorted arrays into a single sorted array.
11. Merge Sort
12. Greatest Common Divisor(GCD) of 2 Numbers
13. Tower of Hanoi
14. Pascals Triangle

Extra

- Prove that weird(n) returns 1 for all positive integers n.

  ```
  public static int weird(int n) {
      if(n==1)
          return 1;
      else if(n%2 == 0)
          return weird(n/2);
      else
          return weird(n+1);
  }
  ```

- It is a famous conjecture in mathematics that the following function weirder(n) returns 1 for all positive integers n. No one has been able to prove it so far. Simplelooking recursive functions may exhibit complex behavior.

```
public static int weirder(int n) {
    if(n==1)
        return 1;
    else if(n%2 == 0)
        return weirder(n/2);
    else
        return weirder(3*n+1);
}
```