# LAB #11 - RECURSION

## PART 1: Exercises

**1.** Add whatever necessary to fix the following method so that `f(3)` equals 10.

```
public static int f(int n) {
    return f(n - 1) + 3;
}
```

---

**2.** If the following method is called with a value of 73 for `n`, what is the resulting output?

```
public static void f(int n) {
     if (n > 0) {
         f(n / 5);
         System.out.print(n % 5);
     }
 }
```

---

**3.** In problems 1 - 3, show the output that will be displayed by the call `f(123)`;

```
1. public static void f(int n) {
       if (n != 0)
           f(n / 10);
       System.out.print(n % 10);
   }

2. public static void f(int n) {
       System.out.print(n % 10);
       if (n != 0)
           f(n / 10);

   }

3. public static void f(int n) {
       System.out.print(n % 10);
       if (n != 0)
           f(n / 10);
       System.out.print(n % 10);
   }
```

---

**4.** Given the following recursive method:

```
public static int f(int num) {
    if(num == 0)
        return 0;
    else
        return num + f(num + 1);
}
```

1. Is there a constraint on the values that can be passed as parameters for this function?
2. Is `f(7)` a good call? If so, what is returned from the function?

3. Is `f(0)` a good call? If so, what is returned from the function?
4. Is `f(-5)` a good call? If so, what is returned from the function?

---

**5.** Given the recursive method (base and limit are nonnegative numbers):

```
public static int f(int base, int limit){
    if(base > limit)
        return -1;
    else if (base == limit)
        return 1;
    else
        return base * f(base + 1, limit);
}
```

a) Identify the base case(s) for this method.
b) Identify the recursive case(s) for this method.
c) Show what would be displayed by the following calls.

```
System.out.println(f(14,10));
System.out.println(f(4,7));
System.out.println(f(0,0));
```

---

**6.** Consider the method below:

```
public static void f(char param1, char param2) {
     if (param1 != param2) {
         param1++;
         param2--;
         f(param1, param2);
         System.out.print(param1);
         System.out.print(param2);
      }
}
```

Show all output resulting from the method call `f('A', 'G');`

---

**7.** Given the following recursive method:

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

1. What problems come up in verifying this method?
2. How many recursive calls are made by the following initial calls?

```
System.out.println(f(7));
System.out.println(f(8));
System.out.println(f(15));
```

```
System.out.println(f(16));
```

---

**8.** Given the following recursive method:

```
public static int f(int num) {
    if(num == 1)
        return 0;
    else if(num % 2 == 1)
        return 1 + f(3 * num + 1);
    else
        return 100 + f(num / 2);
}
```

What is returned from the method after the call for `f(3)`?

---

**9.** Given the following recursive method:

```
public static int f(int x, int y) { // x > y
    if(x % y == 0)
        return y;
    else
        return f(y, x % y);
}
```

What is returned from the method after the call for `f(36, 16)`?

---

**10.** Given the following recursive method:

```
public static int f(int num) {
    if(num != 5)
        return 3 + f(num + 1);
    else
        return 5;
}
```

What is returned from the method after the call for `f(1)`?

---

**11.** Given the following recursive method:

```
public static void f(int x) {
    if(x > 0 && x < 10){
        System.out.print(x + " ");
        f(x + 1);
    }
}
```

What will be the output for the calls: `f(0), f(5), f(10), and f(-5)`?

---

**12.** Given the following recursive method:

```
public static int f(int x, int y) {
    if(x == y)
        return x;
    else if (x > y)
```

```
            return x + y;
        else
            return f(x + 1, y - 1);
    }
```

What will be the output for the calls: `f(5, 10), and f(3,9)`?

---

**13.** Given the following recursive method:

```
    public static int f(int x, int y) {
        if(x < y)
            return x;
        else
            return f(x - y, y);
    }
```

For each call below, indicate the value returned: `f(7, 10), f(16, 2), f(15, 9), f(27, 4), f(59, 10)`

This method does.... what?

---

**14.** Given the following recursive method:

```
    public static void f(int x) {
        if(x <= 1)
            System.out.print(x + " ");
        else {
            f(x / 2);
            System.out.print(x + " ");
        }
    }
```

What will be the output for the call: `f(7)`?

---

## PART 2: Programming

Write a program to do the testing for the following methods. Make sure you also have a method to handle input validation (positive integer, `n > 0`).

1. A method to find the sum of digits in any integer (iterative method)
2. A method to find the sum of digits in any integer (recursive method)
3. A method to display a number in binary (iterative method)
4. A method to display a number in binary (recursive method)
5. A method to return the string representation of a number in any base (iterative method)
6. A method to return the string representation of a number in any base (recursive method)

### SAMPLE OUTPUT:

```
Enter a positive integer: weqrwer
WRONG TYPE! Not a positive integer! REENTER: 123.45
WRONG TYPE! Not a positive integer! REENTER: -12345
INVALID! Should be positive! REENTER: 1024
Enter a positive integer for base: 2
Sum of digits for 1024 is 7 (iterative solution)
Sum of digits for 1024 is 7 (recursive solution)
1024 in binary code is 10000000000(iterative solution)
```

```
1024 in binary code is 10000000000(recursive solution)
1024 in base 2 is 10000000000 (iterative solution)
1024 in base 2 is 10000000000 (recursive solution)
Do you want to continue (y/Y): y
Enter a positive integer: 256
Enter a positive integer for base: 4
Sum of digits for 256 is 13 (iterative solution)
Sum of digits for 256 is 13 (recursive solution)
256 in binary code is 100000000(iterative solution)
256 in binary code is 100000000(recursive solution)
256 in base 4 is 10000 (iterative solution)
256 in base 4 is 10000 (recursive solution)
Do you want to continue (y/Y): y
Enter a positive integer: 123456789
Enter a positive integer for base: 1234
Sum of digits for 123456789 is 45 (iterative solution)
Sum of digits for 123456789 is 45 (recursive solution)
123456789 in binary code is 111010110111100110100010101(iterative solution)
123456789 in binary code is 111010110111100110100010101(recursive solution)
123456789 in base 1234 is 819225 (iterative solution)
123456789 in base 1234 is 819225 (recursive solution)
Do you want to continue (y/Y): y
Enter a positive integer: 257
Enter a positive integer for base: 4
Sum of digits for 257 is 14 (iterative solution)
Sum of digits for 257 is 14 (recursive solution)
257 in binary code is 100000001(iterative solution)
257 in binary code is 100000001(recursive solution)
257 in base 4 is 10001 (iterative solution)
257 in base 4 is 10001 (recursive solution)
Do you want to continue (y/Y): n
```