Exercise 1

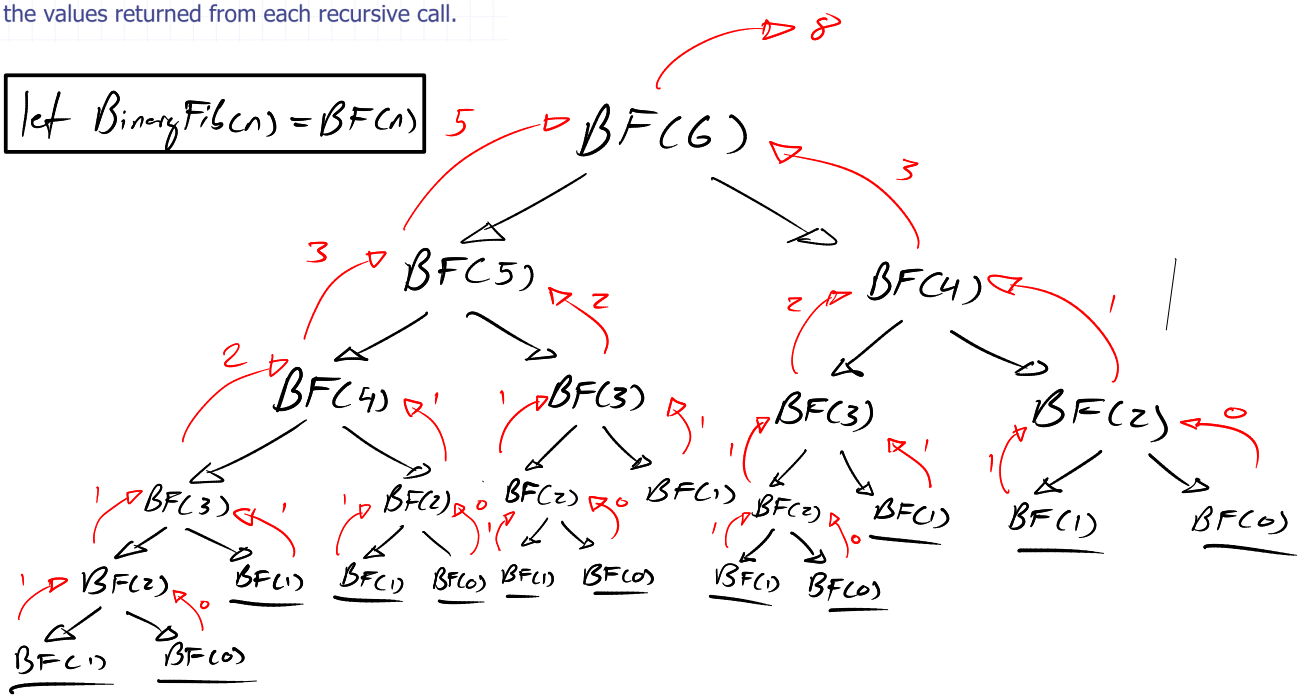
- Draw the recursion trace of the call BinaryFib(6).
- Draw the recursion trace of the call LinearFib(6).

Show the values returned from each recursive call.

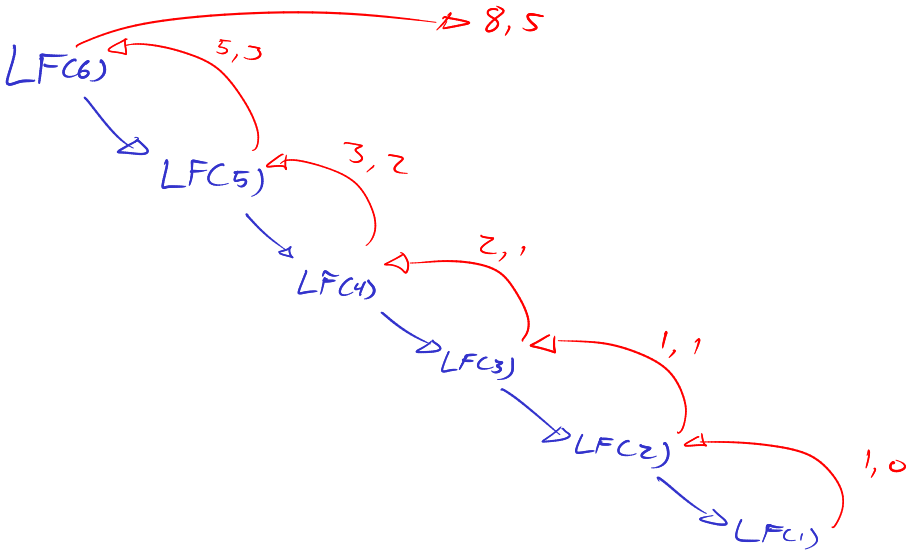
Aaron Patrick Monte
k20059926

4CCS1DST - Worksheet 1

let BinaryFib(n) = BF(n)



Let LinearFib(n) = LF(n)



Exercise 2

- Implement the recursive and non-recursive (iterative) methods for reversing an array.

  That is, write Java methods which implement the algorithms:

  ReverseArray(*A, i, j*)

  IterativeReverseArray(*A, i, j*)

ReverseArray - Week 1 - Programming

Class Edit Tools Options

ReverseArray

Compile Undo Cut Copy Paste Find... Close Source Code

```java
/**
 * ReverseArray
 *
 * @author Aaron Patrick Monte
 * @version 24.01.2021
 */
public class ReverseArray
{
    // instance variables - replace the example below with your own
    private int[] x = {5, 8, 9, 3, 6, 1};
    private int i = 0;
    private int j = x.length - 1;

    /**
     * Constructor for objects of class ReverseArray
     */
    public ReverseArray(){
    }

    /**
     * Output the array
     */
    public void printArray()
    {
        for (int i = 0; i < x.length; i++) {
            System.out.println(x[i]);
        }
    }

    /**
     * Reverse the array using recursion
     */
    public void reverseArray(){
        int temp; // Used to temporarily store the value of x[i] to perform the swap.

        if (i < j) {
            temp = x[i];
            x[i] = x[j];
            x[j] = temp;

            i = i + 1; //increment values
            j = j - 1;

            reverseArray();
        }
    }
}
```

saved

Exercise 2

- Implement the recursive and non-recursive (iterative) methods for reversing an array.

  That is, write Java methods which implement the algorithms:

  ReverseArray(*A, i, j*)

  IterativeReverseArray(*A, i, j*)

IterativeReverseArray - Week 1 - Programming

Class Edit Tools Options

ReverseArray | IterativeReverseArray

Compile Undo Cut Copy Paste Find... Close Source Code

```java
/**
 * IterativeReverseArray
 *
 * @author Aaron Patrick Monte
 * @version 24.01.2021
 */
public class IterativeReverseArray
{
    // instance variables - replace the example below with your own
    private int[] x = {5, 8, 9, 3, 6, 1};
    private int j = x.length - 1;

    /**
     * Constructor for objects of class IterativeReverseArray
     */
    public IterativeReverseArray(){
    }

    /**
     * Output the array
     */
    public void printArray()
    {
        for (int i = 0; i < x.length; i++) {
            System.out.println(x[i]);
        }
    }

    /**
     * Reverse the array using recursion
     */
    public void reverseArray(){
        int temp; // Used to temporarily store the value of x[i] to perform the swap.

        for (int i = 0; i < x.length; i++) {
            if (i < j) {
                temp = x[i];
                x[i] = x[j];
                x[j] = temp;

                j = j - 1;
            }
        }
    }
}
```

Class compiled - no syntax errors

saved

Exercise 3

- Consider the sequence of numbers P_0, P_1, P_2, P_3 … defined recursively:

  $P_0 = P_1 = P_2 = 1$,

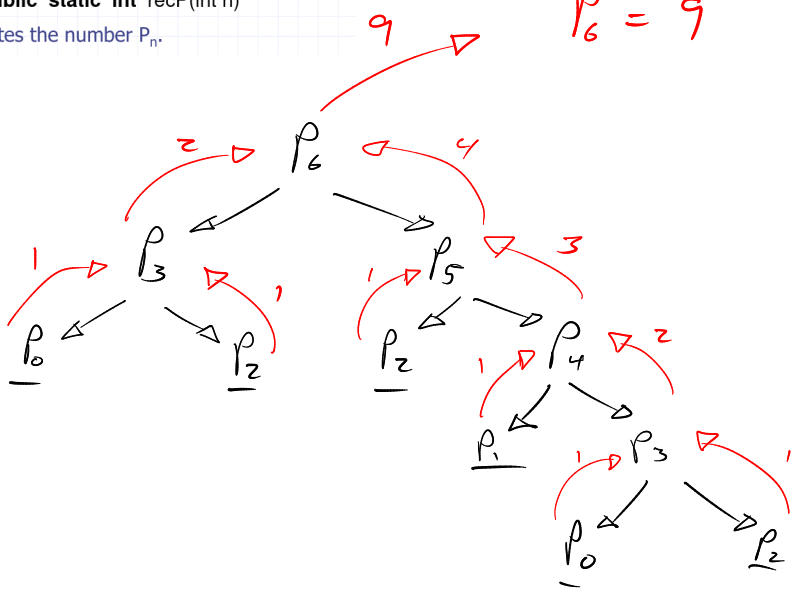
  $P_n = (P_{n-3} * P_{n-1}) + 1$, for $n \geq 3$.

- Calculate P_6
- Write a straight recursive Java method

  public static int recP(int n)

  which computes the number P_n.

$P_6 = 9$



Exercise 3

- Consider the sequence of numbers P_0, P_1, P_2, P_3 … defined recursively:

$$P_0 = P_1 = P_2 = 1,$$
$$P_n = (P_{n-3} * P_{n-1}) + 1, \text{ for } n \geq 3.$$

- Calculate P_6
- Write a straight recursive Java method

```
public static int recP(int n)
```

which computes the number P_n.

RecP - Week 1 - Programming

Class Edit Tools Options

ReverseArray | IterativeReverseArray | RecP

Compile Undo Cut Copy Paste Find... Close Source Code

```java
/**
 * RecP
 *
 * @author Aaron Patrick Monte
 * @version 24.01.2021
 */
public class RecP
{
    // instance variables - replace the example below with your own

    /**
     * Constructor for objects of class recP
     */
    public RecP()
    {
    }

    /**
     * recP
     */
    public int recP(int n)
    {
        if (n < 3) {
            return 1;
        }
        else {
            return (recP(n-3) * recP(n-1)) + 1;
        }
    }
}
```

saved

Exercise 3 (cont.)

- Consider the computation of recP(8), including all calls at all levels of recursion. How many calls recP(4) are there during this computation?
  Justify your answer by drawing the relevant part of the recursion tree.

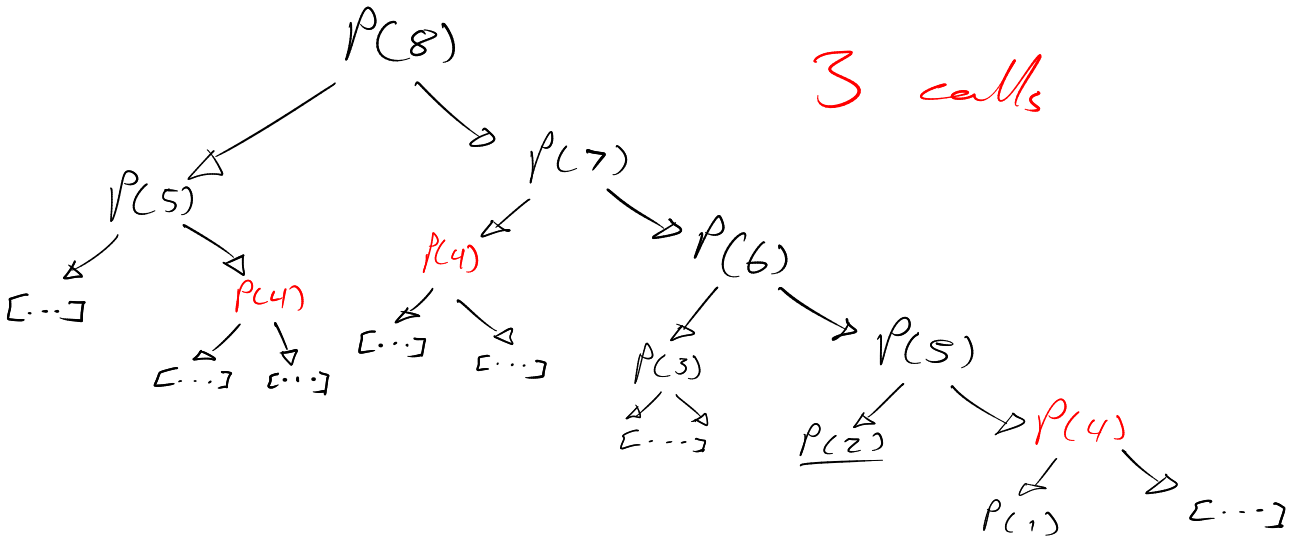
- Write an iterative (non-recursive) Java method

```
public static int iterP(int n)
```

  which computes the number P_n.

$P_0 = P_1 = P_2 = 1,$

$P_n = (P_{n-3} * P_{n-1}) + 1$, for $n \geq 3$.



3 calls