LogicVis

Candice Miao
Leo Gao
Jed Chen
Glenn Zhang
Andrew Liu

Recursion is one of the **top 3** most difficult questions.

Elliott Tew, Allison & Guzdial, Mark. (2011). The FCS1: A language independent assessment of CS1 knowledge. SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. 10.1145/1953163.1953200.

Facilitate the learning experience of recursion

for new recursion learners

Previous Approach

Code Visualization Tools

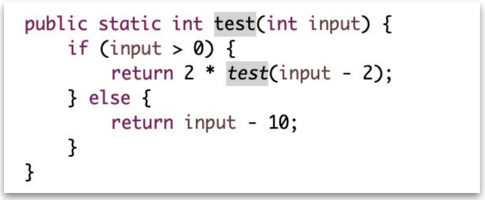
- Eclipse CFG Generator
  - No indication of recursion or even function call

Recursion Trackers

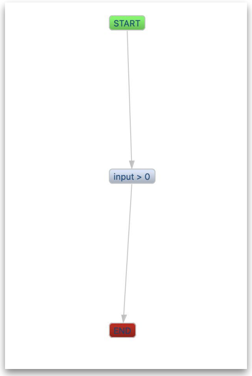
- VisuAlgo
  - No detail about the function is presented

ChiQat-Tutor system

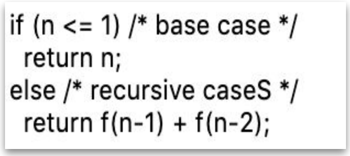
- A system that helps visualize the recursive calls
- Only visualizes pre-defined cases



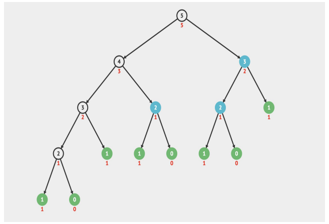
```
public static int test(int input) {
    if (input > 0) {
        return 2 * test(input - 2);
    } else {
        return input - 10;
    }
}
```



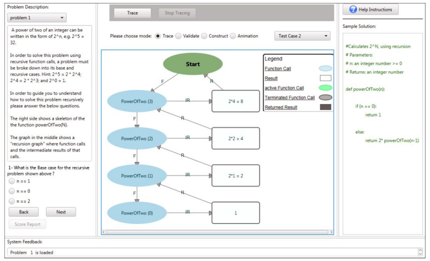
(Eclipse CFG Generator)



```
if (n <= 1) /* base case */
  return n;
else /* recursive caseS */
  return f(n-1) + f(n-2);
```



(VisuAlgo)



Problem Description:

problem 1

A power of two of an integer can be written in the form of 2^n, e.g. 2^5 = 32.

In order to solve this problem using recursive function calls, a problem must be broke down into its base and recursive cases. Hint: 2^5 = 2 * 2^4; 2^4 = 2 * 2^3; and 2^0 = 1.

In order to guide you to understand how to solve this problem recursively please answer the below questions.

The right side shows a skeleton of the the function powerOfTwo(N).

The graph in the middle shows a "recursion graph" where function calls and the intermediate results of that calls.

1- What is the Base case for the recursive problem shown above ?

○ n == 1
○ n == 0
○ n == 2

Please choose mode: [✓ Trace] Validate Construct Animation

Test Case 2

Sample Solution:

```
#Calculates 2^N, using recursion
# Parameters:
# n: an integer number >= 0
# Returns: an integer number

def powerOfTwo(n):

    if (n == 0):
        return 1

    else:
        return 2* powerOfTwo(n-1)
```

System Feedback:

Problem 1 is loaded

(ChiQat-Tutor
)

LogicVis is a program designed

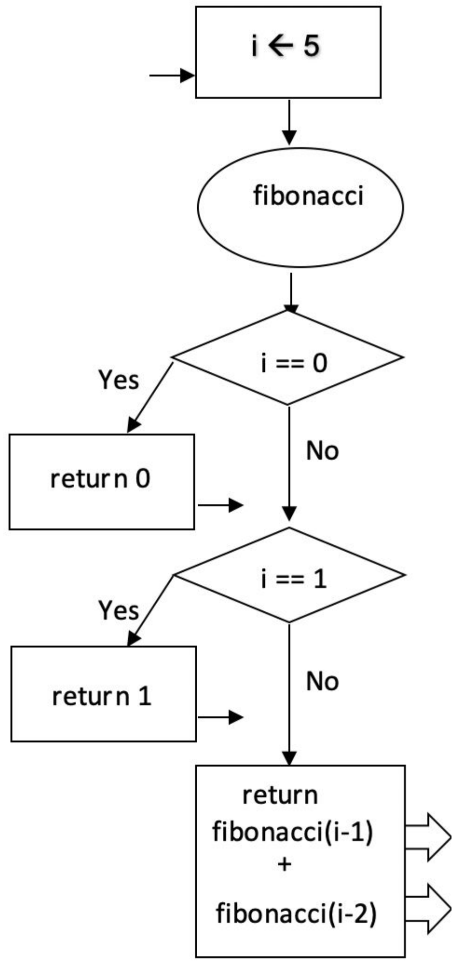
to help recursion beginners better **understand** recursion

by outputting a logic control flow graph

from any recursion method source code provided by the user.

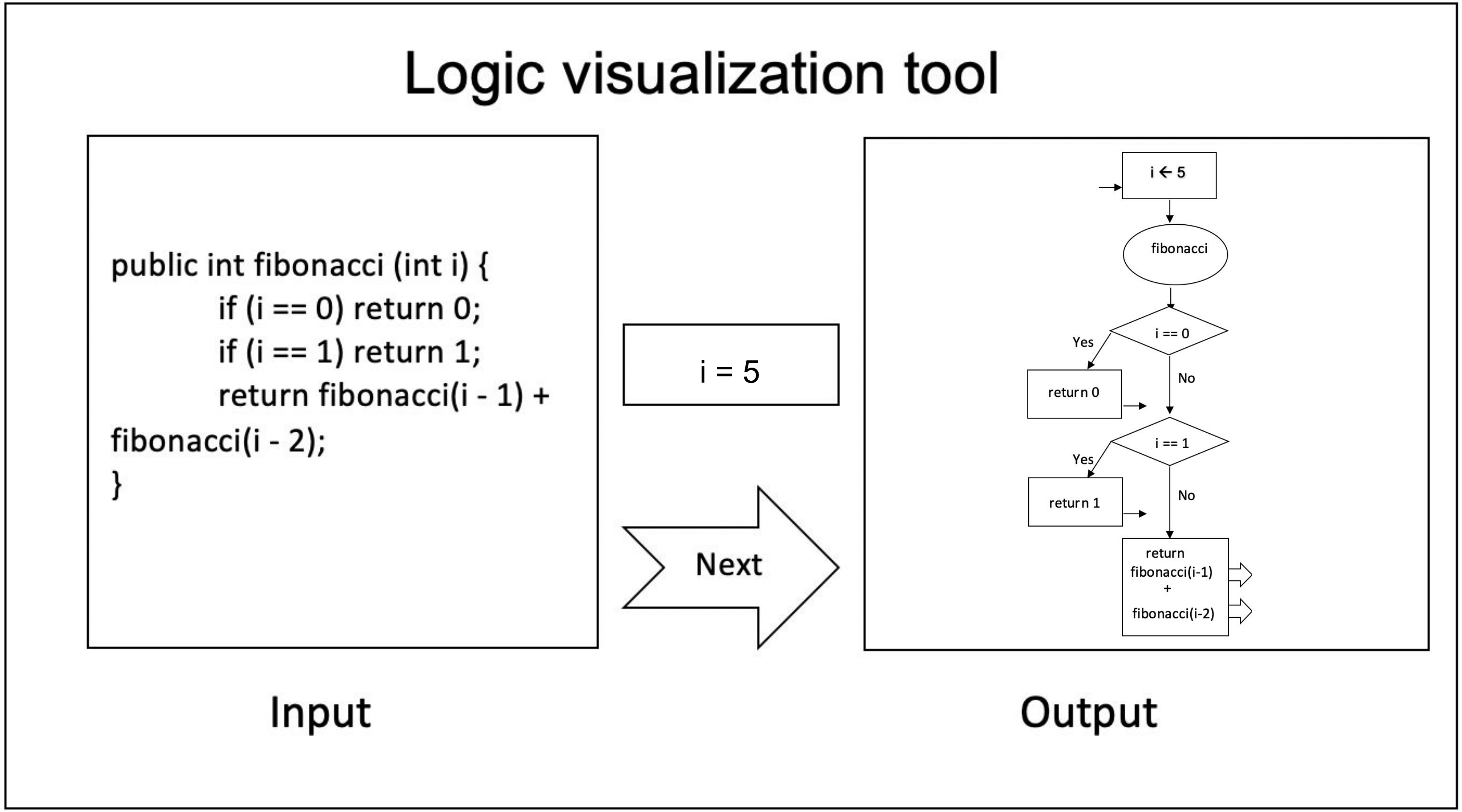
Logic Control Flow Graph

```
public int fibonacci (int i) {
        if (i == 0) return 0;
        if (i == 1) return 1;
        return fibonacci(i - 1) + fibonacci(i - 2);
}
```



“The charts did not help me understand a specific line of code, but they made the recursion execution flow **much easier** to follow and understand”.

Logic visualization tool



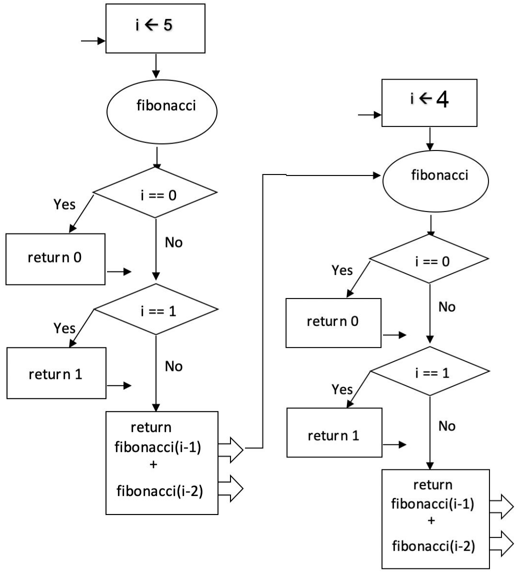
```
public int fibonacci (int i) {
        if (i == 0) return 0;
        if (i == 1) return 1;
        return fibonacci(i - 1) +
fibonacci(i - 2);
}
```

Logic visualization tool

```
public int fibonacci (int i) {
          if (i == 0) return 0;
          if (i == 1) return 1;
          return fibonacci(i - 1) +
fibonacci(i - 2);
}
```

i = 5

Next

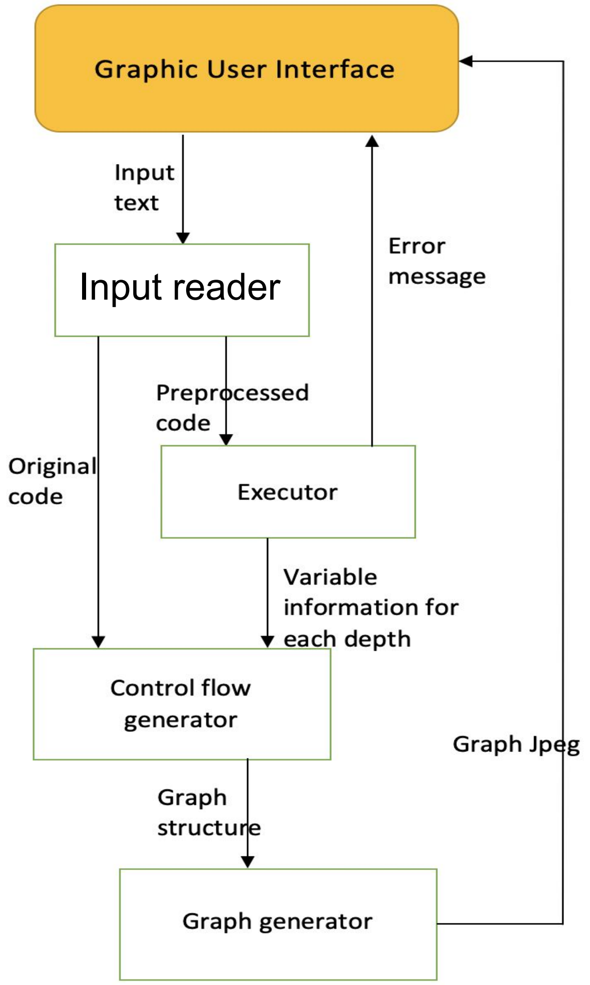


Input

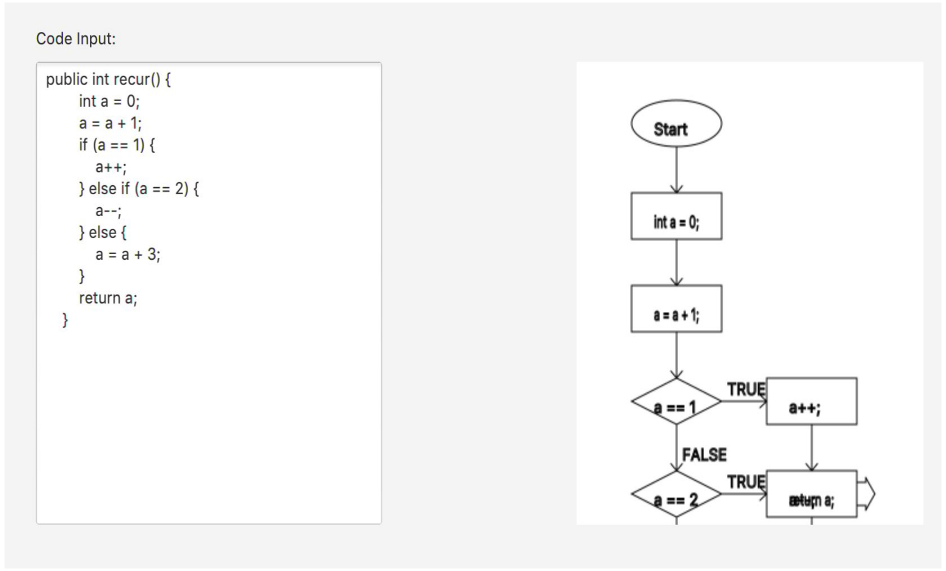
Output

Architecture

- GUI (Front-end)
- Input reader
- Control flow generator
- Executor
- Graphic generator

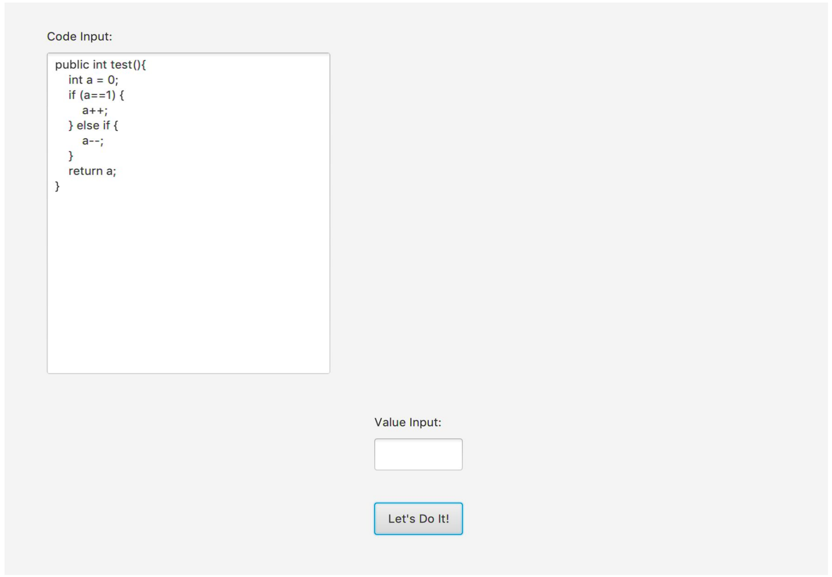


Front-end GUI



Code Input:

```
public int recur() {
      int a = 0;
      a = a + 1;
      if (a == 1) {
         a++;
      } else if (a == 2) {
         a--;
      } else {
         a = a + 3;
      }
      return a;
   }
```



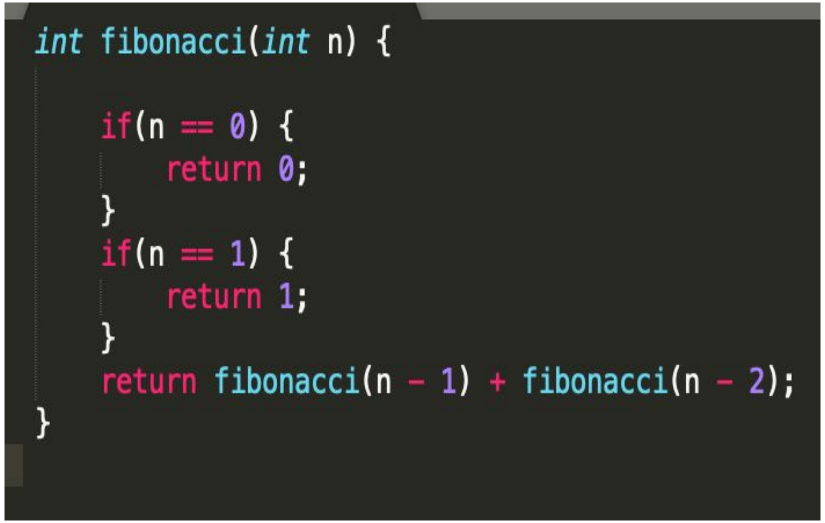
Code Input:

```
public int test(){
   int a = 0;
   if (a==1) {
      a++;
   } else if {
      a--;
   }
   return a;
}
```

Value Input:

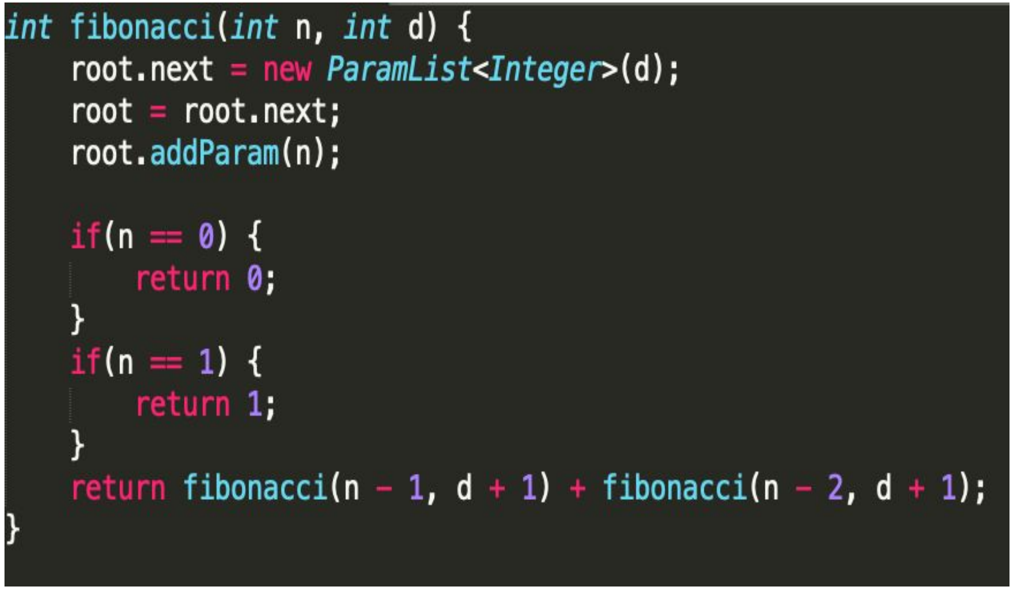
Input reader

1. Read the input → Control flow generator
2. Preprocessed the code → Executor



```
int fibonacci(int n) {

    if(n == 0) {
        return 0;
    }
    if(n == 1) {
        return 1;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

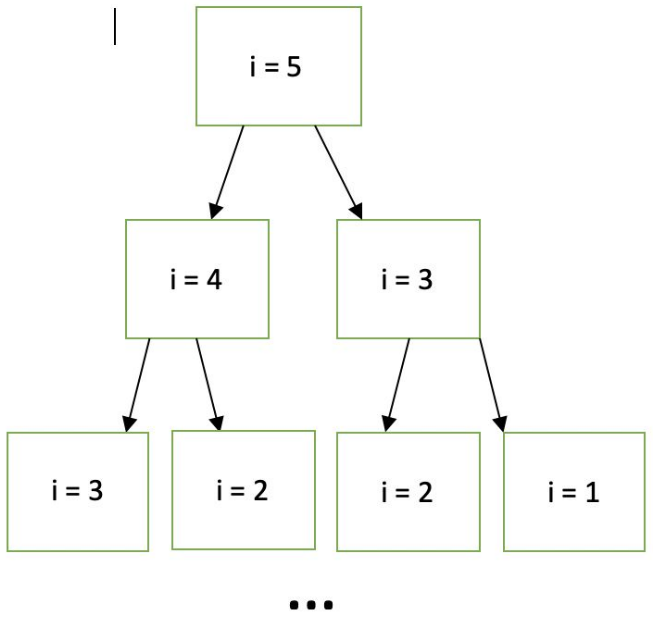


```
int fibonacci(int n, int d) {
    root.next = new ParamList<Integer>(d);
    root = root.next;
    root.addParam(n);

    if(n == 0) {
        return 0;
    }
    if(n == 1) {
        return 1;
    }
    return fibonacci(n - 1, d + 1) + fibonacci(n - 2, d + 1);
}
```

Executor

1. Execute the code, if failed, send error to Front end
2. Execute the code send a tree structure that stores depth information to graph generator
3. Tool : Bean shell http://www.beanshell.org



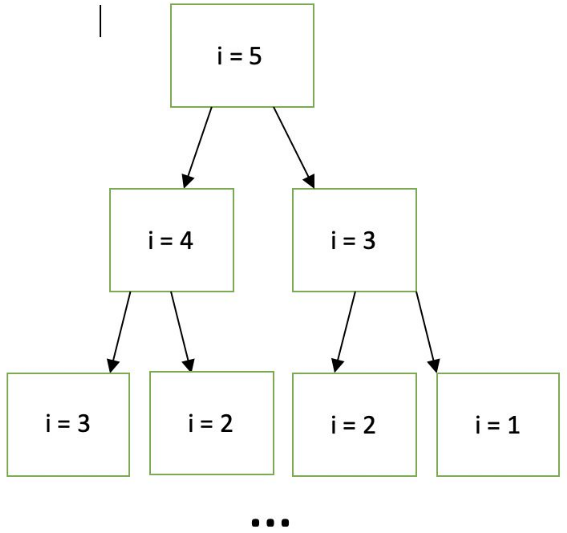
Input Parser / Control Flow Data Structure Generator

- JavaParser API *http://javaparser.org/*
- Hard-code for the types of statements
- Current: if-elseif-else, standard for loop, return

Graph generator

- Visualize the control flow

- *Pre-order traverse the tree and be able to show graph one by one by clicking (Still working)*



User Evaluation Study

“printTwos(int) takes an integer and […]. For example, printTwos(24) should print out “2 * 2 * 3 * 2”. However, the code given does not work properly. Your task is to find and fix the bug.”

```
private void printTwos(int n) {
    if (n % 2 == 0) {
        System.out.print("2 * ");
        n = n / 2;
        printTwos(n);
        if (n % 2 == 0) {
            System.out.print(" * 2");
        }
    } else {
        System.out.print(n);
    }
}
```

User Evaluation Study

“removeOnes(int) takes an integer and [...]. For example, removeOnes(161130) should return the integer 60030. Your task is to fill in the blanks so that the program works as intended.”

```
private int removeOnes(int n) {
    if (____________) {
        return n;
    }
    if (n % 10 == 1) {
        return _______________;
    }
    return ______________;
}
```

Our Measurements

- Redundant errors
- Use of our tool: “calculator” vs. flowchart features

Thank you!