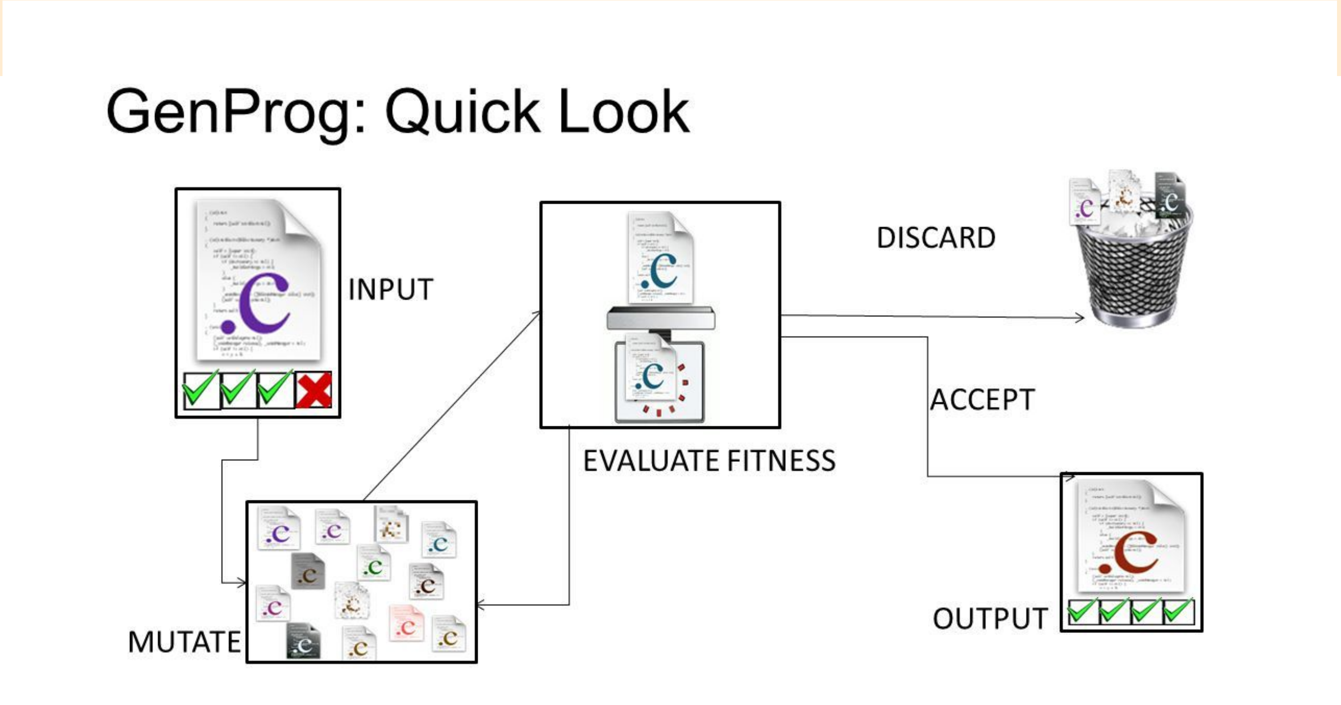
TOOL DEMO
KGenProg

Melisa Panaccione

kGenProg

Genetic Programming



OVERVIEW

TYPE OF TOOL

- Genetic programming
- Successor of GenProg and jGenProg

ALGORITHMS

- Ochiai (fault localization)
- Genetic algorithm (create modified versions)

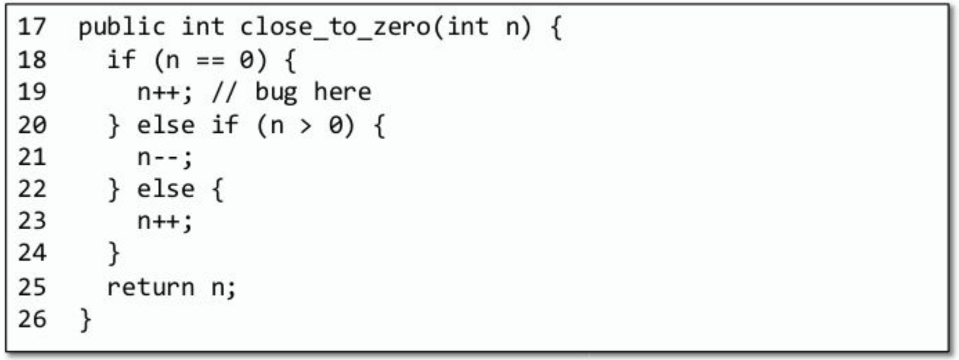
USE

- Repair bugs given a faulty code and a set of test cases

WHY I CHOSE THE TOOL

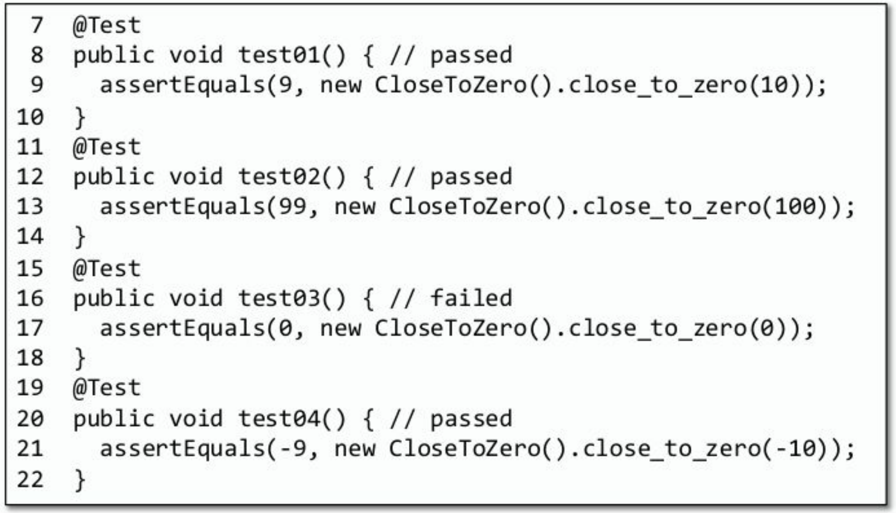
- Curious about the use of GenProg
- Clear setup instructions

Example



```java
public int close_to_zero(int n) {
  if (n == 0) {
    n++; // bug here
  } else if (n > 0) {
    n--;
  } else {
    n++;
  }
  return n;
}
```

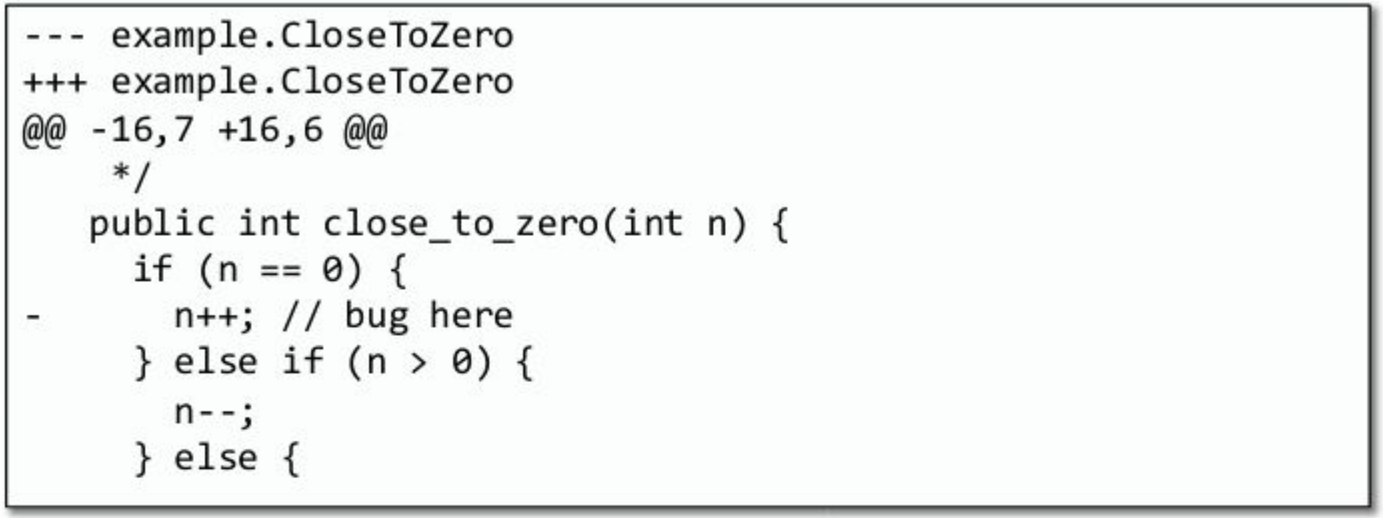
(a) Faulty program



```java
@Test
public void test01() { // passed
  assertEquals(9, new CloseToZero().close_to_zero(10));
}
@Test
public void test02() { // passed
  assertEquals(99, new CloseToZero().close_to_zero(100));
}
@Test
public void test03() { // failed
  assertEquals(0, new CloseToZero().close_to_zero(0));
}
@Test
public void test04() { // passed
  assertEquals(-9, new CloseToZero().close_to_zero(-10));
}
```

(b) Test cases

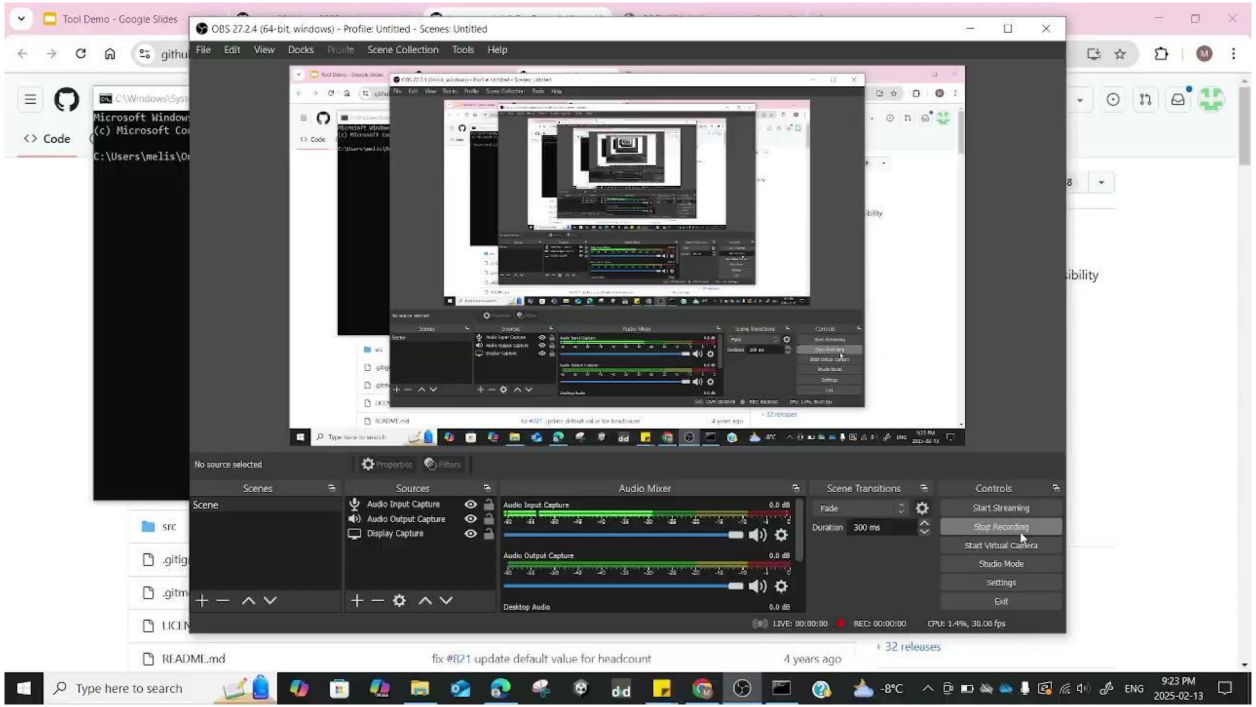
Example



```
--- example.CloseToZero
+++ example.CloseToZero
@@ -16,7 +16,6 @@
    */
   public int close_to_zero(int n) {
     if (n == 0) {
-      n++; // bug here
     } else if (n > 0) {
       n--;
     } else {
```

(c) Generated Patch

Demo



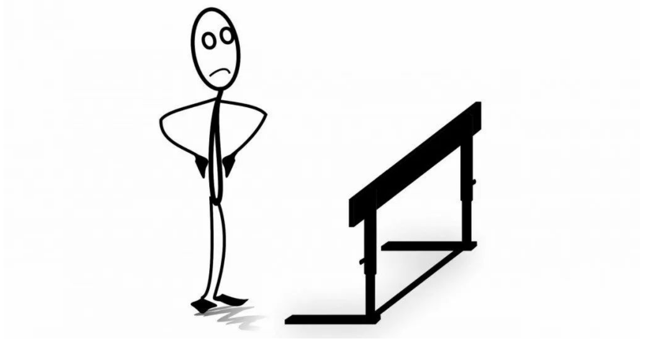
```
Microsoft Window
(c) Microsoft Co

C:\Users\melis\O
```

RQ) How does kGenProg differ from its successor algorithms?

1) In-memory computation
2) Strategy pattern
3) High portability
4) Visualizing the process of fault modification

BIGGEST LIMITATIONS

- Can only add code that already exists in the program
- Constrained to predefined rules
- Can’t fix complex logical errors (no refactoring code)



Thank you