

CSE Research and Technology Forum 2024



THE DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

計算機科學及工程學系



May LLMs tell how my system can go wrong?

Shing-Chi Cheung

What if seeking assistance over a buggy program?

```
Please summarize p1
def p1(a, b):
    if b == 0:
        return a
    else:
        return p1(a, a % b)
```

The function `p1(a,b)` is a recursive function that takes two arguments `a` and `b`. If `b` is equal to 0, the function returns `a`. Otherwise, it calls itself with arguments `a` and `a % b`. This process continues until `b` is equal to 0. The function then returns the value of `a`.



Here is the code for the function:

```
def p1(a, b):
    if b == 0:
        return a
```

```
#correct version
def gcd(a, b):
    if y == 0:
        return a
    else:
        return gcd(b, a % b)
```

The `p1(a,b)` function ultimately returns the greatest common divisor (GCD) of two numbers `a` and `b`. The function uses the Euclidean algorithm to compute the GCD recursively. The Euclidean algorithm is based on the principle that the GCD of two numbers does not change if the larger number is replaced by its remainder when divided by the smaller number. The algorithm continues until the remainder is 0, at which point the GCD has been found.



The intention of the `p1(a,b)` function is to find the greatest common divisor (GCD) of two numbers `a` and `b`. The function uses the Euclidean algorithm to compute the GCD recursively. The Euclidean algorithm is based on the principle that the GCD of two numbers does not change if the larger number is replaced by its remainder when divided by the smaller number. The algorithm continues until the remainder is 0, at which point the GCD has been found .



Yes, the `p1(a,b)` function is correct and will return the greatest common divisor (GCD) of two numbers `a` and `b`. The function uses the Euclidean algorithm to compute the GCD recursively. The Euclidean algorithm is a well-known and widely used algorithm for computing the GCD of two numbers.

Can we trust Bing's assessment on our code?



Suggest a test case for p1. What value will be output by p1? What should be the expected output?

Sure! Here is a test case for p1:

Test case: a = 10 b = 25

Expected output: 5

The output of p1 when called with arguments a = 10 and b = 25 will be 5.

Can we trust the evidence given by Bing?

```
IDLE Shell 3.11.3
File Edit Shell Debug Options Window Help
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023,
Type "help", "copyright", "credits" or "license()")
>>> def p1(a, b):
...     if b == 0:
...         return a
...     else:
...         return p1(a, a % b)
...
>>> p1(10,25)
10
>>> |
```

The actual output of p1(10, 25) is 10.



What are the statement and branch coverages after executing p1(0, 0)?

When p1(0, 0) is executed, the function will enter the if statement and return a. The statement coverage of the test case is 100% because both the if and else statements are executed at least once.

The branch coverage of the test case is 50% because only one branch of the if-else statement is executed.

← contradictory

```
def p1(a, b):  
    if b == 0:  
        return a  
    else:  
        return p1(a, a % b)
```

A 100% statement coverage of p1 implies 100% branch coverage



Suggest a failure-inducing test case for p1, indicating its expected output.

Sure! A failure-inducing test case for p1 would be to call the function with arguments $a = 0$ and $b = 0$. The expected output for this test case is undefined because the GCD of 0 and 0 is not well-defined.

Another failure-inducing test case would be to call the function with arguments $a = 10$ and $b = -25$. The expected output for this test case is also undefined because the GCD of two negative numbers is not well-defined.

inconsistency



Suggest a failure-inducing test case for p1, indicating its expected output.

I apologize for the confusion. I just realized that I made a mistake in my previous response. A failure-inducing test case for `p1` would be to call the function with arguments `a = 10` and `b = 0`. The expected output for this test case is 10 because the GCD of any number and 0 is the number itself.

```
def p1(a, b):  
    if b == 0:  
        return a  
    else:  
        return p1(a, a % b)
```

Actually, `p1(10, 0)` is not a failure-revealing test case. It returns 10, which is the same as the expected output.



This implementation is incorrect. Please suggest a correct implementation of GCD.

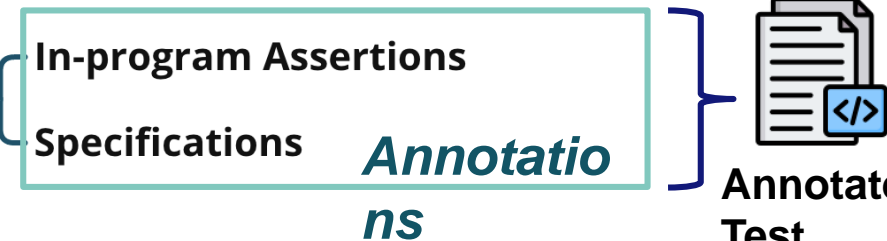
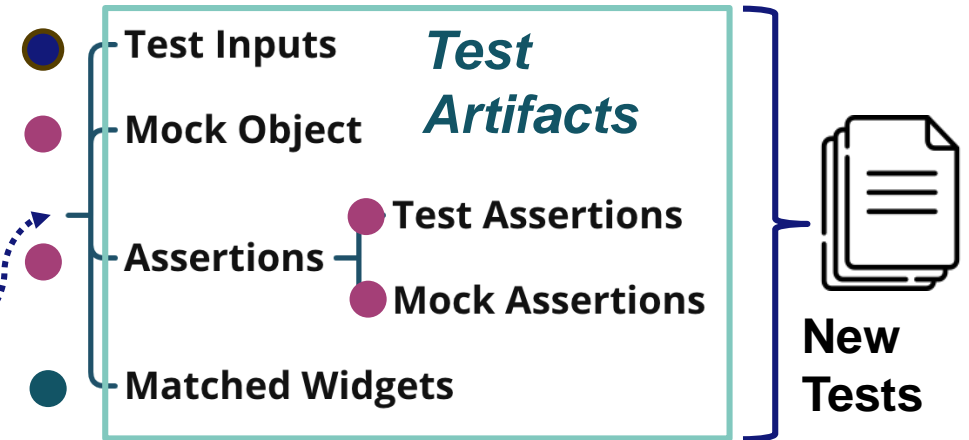
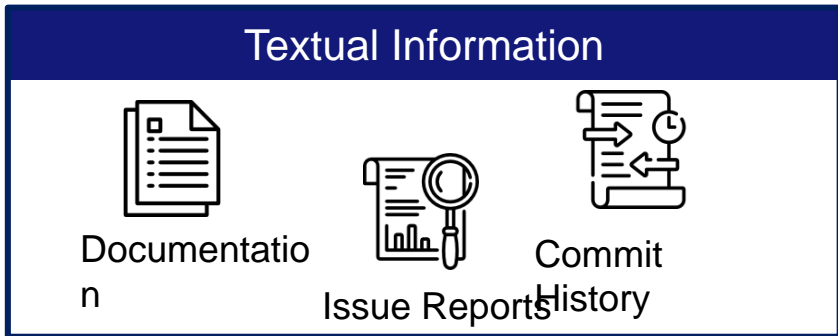
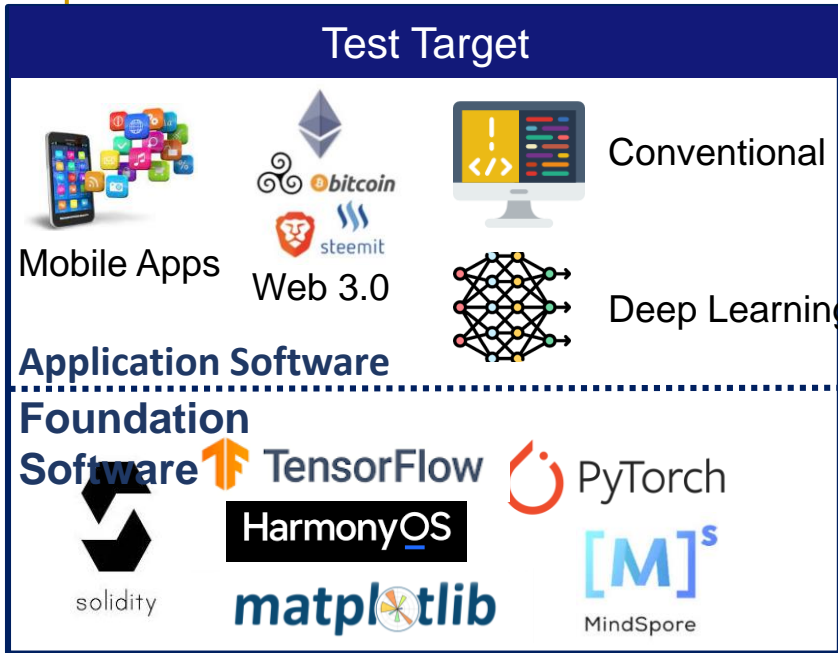
I apologize for the confusion. Here's a correct implementation of the GCD function:

```
def gcd(a, b):  
    if b == 0:  
        return a  
    else:  
        return gcd(b, a % b)
```



This implementation uses the Euclidean algorithm to calculate the GCD of two numbers. The algorithm works by repeatedly taking the remainder of the larger number divided by the smaller number until the remainder is 0. The GCD is then the last non-zero remainder.

Generative Test Framework



Note: We do not differentiate fuzzing from testing

Comparison on Buggy Subjects

	FT-IA	FT-Ia	FT-ia	Passing	Invalid	No Soln	Precision
Pynguin [ASE22]	8%	2%	0%	32%	56%	0%	NA
Direct Use of ChatGPT	29%	19%	8%	30%	15%	0%	43.9%
Differential Prompting	75%	2%	0%	0%	0%	23%	97%

Based on 40 Buggy QuixBugs (before training cutoff)

	FT-IA	FT-Ia	FT-ia	Passing	Invalid	No Soln	Precision
Pynguin [ASE22]	0%	0%	0%	30%	70%	0%	NA
Direct Use of ChatGPT	7%	30%	44%	7%	11%	0%	9%
Differential Prompting	41%	0%	6%	0%	0%	53%	87%

Based on 7 buggy CodeForces (after training cutoff)

Summary

- ChatGPT outperforms state-of-the-art
- Test suites play an important role in using LLMs
- ChatGPT can incorrectly summarize a program based on the code that it has learnt
- Using ChatGPT alone to solve common programming tasks is not effective
- Augmenting ChatGPT with SE techniques can significantly boost its capability in solving common programming tasks