

ML4SE Project

Group A

Ben Tran

Rohit Narayanan

Kendall Hester

Elaina Wittmer

Code Sources

Python

- 50 random functions from CodeSearchNet

Java

- The Pitaya repository
 - Utility library

Python - Executability

Given the Python function below, what is the expected output value (after execution) given <list_of_values>: <Method> Provide no additional explanation, just give me the result. If imports are missing, just import the modules and execute the function with the given values (if any)

Given the Python function below, what is the expected output value (after execution) given mat="[[10], [4]]":

```
def is_square_matrix(mat):
    """Test if an array is a square matrix."""
    mat = np.array(mat)
    if mat.ndim != 2:
        return False
    shape = mat.shape
    return shape[0] == shape[1]
```

Provide no additional explanation, just give me the result. If imports are missing, just import the modules a
=====

Since the `numpy` module is not imported, let's import it before executing the function:

```
```python
import numpy as np

mat="[[10], [4]]"
print(is_square_matrix(mat))
```
```

Expected output: `False`
=====

Actual result:
>>> False

Overall successful - **~76%**
methods correctly reasoned

Mainly on functions that contain concise & clear documentation with relatively short implementation

Python - Executability (cont.)

The expected output value after execution is:

```
'if((3)>0, atan((1)/(3)), if(((3)<0) and ((1)>=0), atan((1)/(3))+3.141592653589793, if(((3)<0) and ((1)<0), atan((1)/(3))-3.141592653589793, if(((3)==0) and ((1)>0), 1.5707963267948966, if(((3)==0) and ((1)<0), -1.5707963267948966, 0))))'
```

Actual result:

```
>>> if((3)>0, atan((1)/(3)), if(((3)<0) and ((1)>=0), atan((1)/(3))+3.141592653589793, if(((3)<0) and ((1)<0), atan((1)/(3))-3.141592653589793, if(((3)==0) and ((1)>0), 3.141592653589793/2, if(((3)==0) and ((1)<0), -3.141592653589793/2, 0))))
```

The expected output value (after execution) is a string:

```
"hnerpiuhf1e90hf03eufneh1f0e89fh1uc89shc8hc81hc0891hec0e89c8h18ech13eerhc8901ecc-jdasi9oud923ud9a2-isud9sud90ajc09aud2309dy029u2n09cuq290u0a09as09"
```

Actual result:

```
>>> hnerpiuhf1e90hf03eufneh1f0e89fh1uc89shc8hc81hc0891hec0e89c8h18ech13eerhc8901eccjdasi9oud923ud9a2-isud9sud90ajc09aud2309dy029u2n09cuq290u0a09as09
```

Most interesting case is chatGPT evaluating a logic expression in a string as a return value

But usual failure cases are misinterpreting the execution, like adding an extra '-' in the result, unable to do calculation, gave a generalized answer, etc.

Python - Executability (cont.)

```
This function is a helper function and cannot be executed on its own.  
It is called by another function and it is not clear what the inputs  
to that function are. Therefore, it is not possible to provide the  
expected output value.
```

```
=====
```

Actual result:

```
>>> True
```

```
Since the input `codes_regex` is not a valid string, it would raise a  
`NameError` when executing the function. Therefore, the expected output  
value cannot be determined without correcting the input value.
```

```
=====
```

Actual result:

```
>>> 5
```

ChatGPT sometimes even assumed the function is not executable, even if it is possible

- A hint of heavy reliance on function documentation

And sometimes it parses the value incorrectly, and assume an error will occur

- Can be attributed to human error in prompt formatting

Java - Executability

Prompt: “Given the Java function below, what is the expected output value (after execution) given <list_of_values>: <Method>”

```
The function `stripLeft` takes a string `str` and an integer `n` as input, and returns a substring of `str` that starts at index `n` (or at the end of the string if `n` is greater than or equal to the length of `str`). If `n` is negative, the function throws an `IllegalArgumentException`. kenetec, last mo
```

```
Assuming the function is called with `str="Don't take this out of context"` and `n="16"`, the expected output value is the string `"out of context"`. Here's why:
```

1. The function first checks that `n` is not negative. In this case, `n` is the string `"16"`, which can be parsed as a non-negative integer, so the check passes.
2. The function then computes `start`, which is the minimum of `str.length()` and `n`. Since `str.length()` is 31 and `n` is 16, `start` is 16.
3. Finally, the function returns the substring of `str` that starts at index 16, which is `"out of context"`.

```
=====
Actual result:
```

```
>>> out of context
```

Overall successful - ~64%
methods correctly reasoned

Mainly on functions manipulating simple primitive types such as booleans and strings or functions that include very descriptive subroutine calls (i.e. URLs).

Java - Executability (cont.)

```
Assuming that n="7523094288207667809" is passed as an argument to the `encode` function, the expected output will be the byte array `out` with 8 bytes starting from index `off`, representing the binary encoding of `n` in little-endian byte order.
```

```
The expected output array will be:
```

```
`[1, 3, 42, -83, 49, -44, 87, 100]`
```

```
=====
```

```
=
```

```
Actual result:
```

```
>>> abcdefgh
```

```
The provided Java function takes two Calendar objects, `cal1` and `cal2`, and compares whether they are representing the same month. The function returns a boolean value, `true` if both calendars represent the same month and `false` otherwise.
```

```
Assuming that the imports are included and the two dates `date1` and `date2` are passed as `Calendar` objects to the function, the expected output value can be determined by executing the following code:
```

```
```java
Calendar date1 = Calendar.getInstance();
date1.set(3924, 11, 10); // Set the year, month, and day of date1
Calendar date2 = Calendar.getInstance();
date2.set(3914, 11, 10); // Set the year, month, and day of date2

boolean result = isSameMonth(date1, date2);
System.out.println(result); // Print the result
```
```

```
The expected output value is `true`, since both dates are in the same month and year (December in the year 3924 and 3914, respectively), and the `isSameMonth` function checks for equality of the `ERA`, `YEAR`, and `MONTH` fields of the Calendar objects.
```

```
=====
```

```
Actual result:
```

```
>>> False
```

ChatGPT failed completely when attempting to reason about any functions with bitwise operations (**LittleEndian**).

It also struggled with functions from the **Dates** class (some errors attributed to representing inputs with a non-native interface).

Python - Unit Testing

“Generate a unit test for the following Python code and provide only the code, no other text: <method>”

```
def test_camel_to_snake_case():
    assert PythonFunctions.camel_to_snake_case('CamelCaseString') == 'camel_case_string'
    assert PythonFunctions.camel_to_snake_case('anotherCamelCaseString') == 'another_camel_case_string'
    assert PythonFunctions.camel_to_snake_case('snake_case_string') == 'snake_case_string'
    assert PythonFunctions.camel_to_snake_case('CONSTANT_CASE_STRING') == 'constant_case_string'
    assert PythonFunctions.camel_to_snake_case('PascalCase') == 'pascal_case'
```

88% tests compiled and executed successfully while only **~61% of those tests** passed.

```
# 100534
def camel_to_snake_case(name):
    """Takes a camelCased string and converts to snake_case."""
    pattern = r'[A-Z][a-z]+|[A-Z]+(?:[a-z])?'
    return '_'.join(map(str.lower, re.findall(pattern, name)))
```

Pain points: Trying to make it create test functions with only one assert statement. The need to provide further context of the functionality of the code to better generate test cases (this tweak resulted in **100% tests** compiling and **72%** of tests passing).

Java - Unit Testing

*“For the following repo: <https://github.com/kocakosm/pitaya>
Using JUnit 4, create a test class and generate a unit test for the following java code: <method>”*

```
24 | @Test
25 | public void testCreateWithNullUrl() {
26 |     assertThrows(NullPointerException.class, () -> {
27 |         URLs.create(null);
28 |     });
29 | }
    |
7 | @Test(expected = NullPointerException.class)
8 | public void testUnquoteNull() {
9 |     Strings.unquote(str:null);
10 | }
```

JUnit 5

JUnit 4

Overall successful - ~70%
tests compiled and
executed successfully

Pain points: Distinguishing
between Junit versions and
handling exceptions

Python - Semantic Equivalency

Consider the following method: <method> Can you generate a different Python method that produces the same output as this one if both are given the same input? Provide no extra explanation.

```
45) Function: merge_dict
Original: {'a': 1, 'c': 4, 'b': 10}
Task 3 : {'a': 1, 'c': 4, 'b': 10}
Similar: True

46) Function: py3round
Original: 10
Task 3 : 11
Similar: False

47) Function: _snake_to_camel_case
Original: ExampleSnakeCase
Task 3 : ExampleSnakeCase
Similar: True

48) Function: deep_copy
Original: {'a': array([1]), 'b': array([2])}
Task 3 : {'a': array([1]), 'b': array([2])}
Similar: True

49) Function: camel_to_snake_case
Original: example_snake_case
Task 3 : example_snake_case
Similar: True

50) Function: stripnull
Original: b'thestringwillhavenullbytehere'
Task 3 : b'thestringwillhavenullbytehere'
Similar: True
```

TOTAL SUCCESS: 36/50

Overall successful - **~72% generated methods** match result from original using the same static input

When running the generated methods on Task 2's test cases, it drops to **~66% methods** passing the test cases

```
def camel_to_snake_case(name):
    """Takes a camelCased string and converts to snake_case."""
    pattern = r'[A-Z][a-z]+|[A-Z]+(?:[a-z])'
    return '_' .join(map(str.lower, re.findall(pattern, name)))

def camel_to_snake_case(name):
    """Takes a camelCased string and converts to snake_case."""
    return ''.join(['_' + c.lower() if c.isupper() else c for c in name]).rstrip('_')
```

Python - Semantic Equivalency (cont.)

Original

```
def _make_safe_pod_id(safe_dag_id, safe_task_id, safe_uuid):
    """
    Kubernetes pod names must be <= 253 chars and must pass the following regex for
    validation
    """
    """^[a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*$"""

    :param safe_dag_id: a dag_id with only alphanumeric characters
    :param safe_task_id: a task_id with only alphanumeric characters
    :param random_uuid: a uuid
    :return: ``str`` valid Pod name of appropriate length
    """

    MAX_POD_ID_LEN = 253

    safe_key = safe_dag_id + safe_task_id

    safe_pod_id = safe_key[:MAX_POD_ID_LEN - len(safe_uuid) - 1] + "-" + safe_uuid

    return safe_pod_id
```

ChatGPT ver.

```
def _make_safe_pod_id(safe_dag_id, safe_task_id, safe_uuid):
    """
    Kubernetes pod names must be <= 253 chars and must pass the following regex for
    validation
    """
    """^[a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*$"""

    :param safe_dag_id: a dag_id with only alphanumeric characters
    :param safe_task_id: a task_id with only alphanumeric characters
    :param random_uuid: a uuid
    :return: ``str`` valid Pod name of appropriate length
    """

    MAX_POD_ID_LEN = 253

    # Use regex to remove any non-alphanumeric characters from safe_dag_id and safe_task_id
    regex = re.compile('[^a-zA-Z0-9]')
    safe_dag_id = regex.sub('', safe_dag_id)
    safe_task_id = regex.sub('', safe_task_id)

    safe_key = safe_dag_id + safe_task_id

    # Truncate safe_key if necessary
    if len(safe_key) > MAX_POD_ID_LEN - len(safe_uuid) - 1:
        safe_key = safe_key[:MAX_POD_ID_LEN - len(safe_uuid) - 1]

    safe_pod_id = safe_key + "-" + safe_uuid

    # Use regex to ensure that the resulting pod id is valid
    regex = re.compile('^[a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*$')
    safe_pod_id = regex.match(safe_pod_id).group()

    return safe_pod_id
```

ChatGPT can be seen relying on the documentation for generating an alternate version of the function

Python - Semantic Equivalency (cont.)

18) FAILURE | function `_splitstrip` has failed for task 3 with error: type object 'Task3_Functions' has no attribute '_splitstrip'

```
def dt_year(x):
    """Extracts the year out of a datetime sample.

    :returns: an expression containing the year extracted from a datetime column.

    Example:

    >>> import yaex
    >>> import numpy as np
    >>> date = np.array(['2009-10-12T03:31:00', '2016-02-11T10:17:34', '2015-11-12T11:34:22'], dtype=np.datetime64)
    >>> df = yaex.from_arrays(date=date)
    >>> df
    #   date
    0 2009-10-12 03:31:00
    1 2016-02-11 10:17:34
    2 2015-11-12 11:34:22

    >>> df.date.dt.year
    Expression = dt_year(date)
    Length: 3 dtype: int64 (expression)
    -----
    0 2009
    1 2016
    2 2015
    """
    import pandas as pd
    return pd.Series(x).dt.year.values
```

```
def dt_year(x):
    """
    Extracts the year out of a datetime sample.

    :returns: an expression containing the year extracted from a datetime column.

    Example:

    >>> date = ['2009-10-12T03:31:00', '2016-02-11T10:17:34', '2015-11-12T11:34:22']
    >>> dt_year(date)
    [2009, 2016, 2015]
    """
    return [datetime.datetime.strptime(date, '%Y-%m-%dT%H:%M:%S').year for date in x]
```

Failures can be seen where ChatGPT can omit '_' at beginning of some functions' signature

ChatGPT sometimes attempt generalized non-builtin imports (Pandas) to more common builtins import (datetime)

- Fails to support the existing input

Constants used tend to not be preserved in the alternate version

Java - Semantic Equivalency

***“For the following repo: <https://github.com/kocakosm/pitaya>
Consider the following method: <method> Can you generate a different Java method that produces the same output as this one? Assume both are given the same input.”***

- 49/50 methods produced – additional 50th generated using new prompt
New prompt included “efficiency does not matter”
- 41/50 methods produced were able to compile
- 36/41 compilable methods were successful
→ Overall 72% success rate

Unsuccessful compilable methods failed due to lack of understanding about object capabilities and exceptions handling

Java - Semantic Equivalency

```
18 public static URL create(String url)
19 {
20     Parameters.checkNotNull(url);
21     try {
22         return new URL(url);
23     } catch (MalformedURLException ex) {
24         throw new IllegalArgumentException(ex);
25     }
26 }
27
```

Original

```
public static URL create(String url) throws MalformedURLException {
    Parameters.checkNotNull(url);
    return new URL(url);
}
```

ChatGPT Version

Java - Semantic Equivalency

***“For the following repo: <https://github.com/kocakosm/pitaya>
Consider the following method: <method> Can you generate a different Java method that produces the same output as this one? Assume both are given the same input.”***

This implementation is simpler because it doesn't catch the `MalformedURLException` thrown by the `URL` constructor and wrap it in an `IllegalArgumentException`. Instead, it simply declares that it may throw the exception, and allows the caller to handle it if necessary. The behavior is the same as the original implementation if the caller doesn't handle the exception.

Python - Data/Control Flow

PRELIMINARY

```
Given the Python function below, what is the expected output
def dt_month(x):
    import pandas as pd
    pdx = pd.Series(x).dt.month
    while pdx.values[0] > 3:
        if pdx.values[0] == 4:
            break
        pdx.values[0] -= 1

    return pd.Series(x).dt.month.values
```

Provide no additional explanation, just give me the result.
If imports are missing, just import the modules and execute
the function with the given values (if any)

=====

The expected output value is an array containing the month
extracted from the datetime values in x, with the first value
modified if it is greater than 3 to be decreased by 1 until
it reaches 4. So for x=["2011-10-12T03:31:00"
'2014-02-11T10:17:34' '2020-11-12T11:34:22']", the expected
output is [10, 2, 11], since none of the values in x have a
month greater than 3.

=====

Actual result:
>>> [10 2 11]

Overall successful

~56% control flow (14/25)

- Good at identifying non-relevant or redundant control flow
- Can sometimes follow relevant flow change, but can still fail

~40% data flow (10/25)

- Harder for ChatGPT to follow

Python - Data/Control Flow (cont.)

PRELIMINARY

```
Given the Python function below, what is the expected return/output value (after
def list_to_string(self, input, delimiter):
    if isinstance(input, list):
        recurse = [self.list_to_string(item, delimiter) for item in input]
        res = delimiter + ' '
        return res.join(recurse)

    return input
```

Provide no additional explanation, just give me the result. If you offer any fi

=====
The expected return value is:

```
'outside, nested1, nested2, finish'
```

=====
Actual result:

```
>>> outside, nested1,nested2, finish
```

CF change on recursion is less susceptible to failure than DF

Doing computation remains a weakness

```
Assuming that numpy has been imported, the expected output after
executing min_max_normalize("[0.001, 0.371, 0.651, 0.861]") is:
```

```
array([0.001      , 0.35294118, 0.62226277, 1.          ])
```

=====
Actual result:

```
>>> [0.00116144 0.43089431 0.75609756 1.          ]
```

Java - Data/Control Flow

Given the Java function below and assuming all imports were included, what is the expected output value (after execution) given values="[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]":

```
public static double mean(long... values)
{
    Parameters.checkCondition(values.length > 0);
    BigDecimal sum = BigDecimal.ZERO;
    for (long value : values) {
        sum = sum.add(BigDecimal.valueOf(value));
    }

    if (Math.abs(sum.longValue()) - Math.abs(sum.longValue()) >= 0) {
        sum = sum.subtract(BigDecimal.valueOf(10));
    }

    sum = sum.add(BigDecimal.valueOf(10));
    return sum.divide(BigDecimal.valueOf(values.length)).doubleValue();
}
```

The expected output value of this function given `values="[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]"` is 28.5.

The `mean` function calculates the average of the input values. In this implementation, it first sums up all the input values using `BigDecimal` to ensure accurate arithmetic. Then it checks if the absolute value of the sum minus itself is greater than or equal to zero. This condition always evaluates to `false`, so it does not affect the result.

Finally, the function adds 10 to the sum, and then divides it by the length of the input array to calculate the average. For the given input values, the sum is 285, so the average is 28.5.

CF: 17 out of 25 functions (68%) are able to be reasoned

- Sometimes ChatGPT ignores the control flow change or evaluates it incorrectly

Java - Data/Control Flow

```
public static <T> T[] rotate(T[] a, int distance)
{
    int len = a.length;
    T[] copy = newArray(a.getClass().getComponentType(), len);
    for (int i = 0; i < len; i++) {
        int idx = index(i - distance, len);
        T x = a[idx];
        T y = x.clone();
        copy[i] = y;
    }
    return copy;
}
```

You, yesterday * add dataflow results

=====

Assuming the implementation of the `index` method is not provided, the expected output value cannot be determined without knowing the implementation of the `index` method.

The `rotate` method takes an array `a` of type `T` and an integer `distance`, and returns a new array `copy` that is a rotation of the input array by `distance` positions.

The `index` method is used to compute the index of the rotated element. The implementation of this method is not provided, so it is impossible to determine the expected output value without knowing how this method works.

Assuming the `index` method is correctly implemented, the expected output value would be a new array `copy` that is a rotation of the input array `a` by `distance` positions.

DF: 16 out of 25 functions (64%) are able to be reasoned

- Sometimes ChatGPT will focus on a function it never had before.

Python - Bug Prediction/Localization

PRELIMINARY

“Given the following testcase: <Test Case Code> Is the following Python code buggy or correct? <Mutated Code>”

| | ChatGPT Reports Bug | ChatGPT Reports Correct |
|----------|---------------------|-------------------------|
| Survived | 9 (7.2%) | 4 (3.2%) |
| Killed | 82 (65.6%) | 30 (24.0%) |

ChatGPT agreed with unit test output **68.8%** of the time

Python - Bug Prediction/Localization

PRELIMINARY

“Given the following testcase: <Test Case Code> Is the following Python code buggy or correct? <Mutated Code>”

Original

```
1 def py3round(number):
2     """Unified rounding in all python versions."""
3     if abs(round(number) - number) == 0.5:
4         return int(2.0 * round(number / 2.0))
5
6     return int(round(number))
```

ChatGPT agreed with unit test output **68.8%** of the time

Python - Bug Prediction/Localization

PRELIMINARY

“Given the following testcase: <Test Case Code> Is the following Python code buggy or correct? <Mutated Code>”

Modified

```
1 def py3round(number):
2     """Unified rounding in all python versions."""
3     if abs(round(number) - number) == 0.5:
4         return int(2.0 // round(number / 2.0))
5
6     return int(round(number))
```

ChatGPT agreed with unit test output **68.8%** of the time

Python - Bug Prediction/Localization

PRELIMINARY

“Given the following testcase: <Test Case Code> Is the following Python code buggy or correct? <Mutated Code>”

The following Python code is buggy.

The problematic statement is `int(2.0 // round(number / 2.0))`. The `//` operator performs integer division, so `2.0 // round(number / 2.0)` will always be either 2.0 or 0.0.

To fix the bug, we should use the `*` operator instead of the `//` operator: `int(2.0 * round(number / 2.0))`. This will correctly compute the rounded value.

ChatGPT agreed with unit test output **68.8%** of the time

In this case, ChatGPT is able to find what is causing the bug, provide reasoning, and even explain what fix will solve it, which in this case is correct.

Python - Bug Prediction/Localization

PRELIMINARY

“The following Python code is buggy. Can you spot the statements involved in the bug?”

| | ChatGPT Correctly Located Bug | ChatGPT Did Not Find (Correct) Bug |
|----------|-------------------------------|------------------------------------|
| Survived | 9 (7.2%) | 4 (3.2%) |
| Killed | 102 (81.6%) ↑ | 10 (8.0%) ↓ |

ChatGPT agreed with unit test output **84.8%** of of the time – Much higher than prediction

In general, ChatGPT was able to pinpoint errors and even provide solutions that match the original (word for word, sometimes) much effectively for Python.

Python - Bug Prediction/Localization

PRELIMINARY

“The following Python code is buggy. Can you spot the statements involved in the bug?”

There is no bug in the code. However, the function is not doing anything and always returns `None`. The `pass` statement is a placeholder statement in Python that does nothing. It is often used as a placeholder when writing code that will be filled in later.

There were times where the code modification will pass through the compiler, and ChatGPT will state there is not bug. However, it will be able to determine for a number of cases like this if there's missing implementation.

Java - Bug Prediction/Localization

PRELIMINARY

“Using the code and the unit test provided, can you determine whether the following Java code is buggy or correct?”

| | ChatGPT Reports Bug | ChatGPT Reports Correct |
|----------|---------------------|-------------------------|
| Survived | 6 (4.8%) | 14 (11.2%) |
| Killed | 66 (52.8%) | 33 (26.4%) |

ChatGPT agreed with unit test output **64%** of of the time

Java - Bug Prediction/Localization

PRELIMINARY

“Using the code and the unit test provided, can you determine whether the following Java code is buggy or correct?”

Original

```
14 public static Boolean and(Boolean... bools)
15 {
16     Parameters.checkCondition(bools.length > 0);
17     for (Boolean bool : bools) {
18         if (!bool) {
19             return FALSE;
20         }
21     }
22     return TRUE;
23 }
```

ChatGPT agreed with unit test output **64%** of of the time

Surprising Finding:
ChatGPT recognized when code had been removed

Java - Bug Prediction/Localization

PRELIMINARY

“Using the code and the unit test provided, can you determine whether the following Java code is buggy or correct?”

Modified

```
14 public static Boolean and(Boolean... bools)
15 {
16 |
17     for (Boolean bool : bools) {
18         if (!bool) {
19             return FALSE;
20         }
21     }
22     return TRUE;
23 }
```

ChatGPT agreed with unit test output **64%** of of the time

Surprising Finding:
ChatGPT recognized when code had been removed

Java - Bug Prediction/Localization

PRELIMINARY

“Using the code and the unit test provided, can you determine whether the following Java code is buggy or correct?”

The `Parameters.checkCondition(bools.length > 0);`` statement in the original code was meant to handle this case, but it was removed in the modified code. Therefore, the modified code is buggy. To make it correct, we can add back the check for the empty input array.

ChatGPT agreed with unit test output **64%** of of the time

Surprising Finding:
ChatGPT recognized when code had been removed

Java - Bug Prediction/Localization

PRELIMINARY

“The following Java code is buggy. Can you spot the statements involved in the bug?”

| | ChatGPT Correctly Located Bug | ChatGPT Did Not Find (Correct) Bug |
|----------|-------------------------------|------------------------------------|
| Survived | 11 (8.8%) ↑ | 9 (7.2%) ↓ |
| Killed | 72 (57.6%) ↑ | 27 (21.6%) ↓ |

ChatGPT agreed with unit test output **64.8%** of of the time – Slightly higher than prediction

In general, ChatGPT reported fewer correct bugs than in the prediction stage

Java - Bug Prediction/Localization

PRELIMINARY

“The following Java code is buggy. Can you spot the statements involved in the bug?”

```
18 public static Date parse(String date, Locale locale, String... formats)
19 {
20     Parameters.checkNotNull(date); conditional boundary change
21     Parameters.checkCondition(formats.length >= 0);
22     for (String format : formats) {
23         DateFormat df = new SimpleDateFormat(format, locale);
24         df.setLenient(true);
25         ParsePosition position = new ParsePosition(0);
26         Date d = df.parse(date, position);
27         if (d != null && position.getIndex() == date.length()) {
28             return d;
29         }
30     }
31     throw new IllegalArgumentException("Unparseable date: " + date);
32 }
```

ChatGPT sometimes reported that the bug was in the documentation or in the unit tests, rather than the code

Java - Bug Prediction/Localization

PRELIMINARY

“The following Java code is buggy. Can you spot the statements involved in the bug?”

There is a **minor bug in the method's Javadoc**. Specifically, the description for the ``formats`` parameter incorrectly indicates that the length of the array must be greater than zero. **It should instead indicate that the length must be greater than or equal to zero**, which matches the condition checked by the ``Parameters.checkCondition`` method.

ChatGPT sometimes reported that the bug was in the documentation or in the unit tests

Thank You