Prompt 1: Write a Python method def cycpattern_check(a , b) for following problem:

You are given 2 words. You need to return True if the second word or any of its rotations is a substring in the first word.

```
# Generated code by GPT_4:
def cycpattern_check(a,b):
   extended_a = a + a    ->   Buggy line
   return any(b[i:] + b[:i] in extended_a for i in range(len(b)))

assert cycpattern_check("xyzw","xyw")==False  -> Failure
```

GPT-4 cannot clearly understand if it should check the existence of “b” and its rotations in “a” or any rotation of “a.” As a result, it generates an incorrect code doing the latter, which causes assertation failure. After including the test in the specification, it is clear for the model that it should only compare “b” and its rotations with only “a.” Hence, it creates a correct code.

Prompt 2: Write a Python method def cycpattern_check(a , b) for following problem:

You are given 2 words. You need to return True if the second word or any of its rotations is a substring in the first word. Your solution should satisfy the following assertion:

assert cycpattern_check("xyzw", "xyw") == False

```
# Generated code by GPT_4:
def cycpattern_check(a, b):
   for i in range(len(b)):
       rotated_b = b[i:] + b[:i]
       if rotated_b in a:
           return True
   return False
assert cycpattern_check("xyzw", "xyw") == False  -> Pass
assert  cycpattern_check("yello","ell") == True  -> Pass
```

While GPT-4 was able to generate the correct code that passed all tests, it cannot correctly reason about the tests that its generated code passes.

Prompt 3: Consider the following code

```
def cycpattern_check(a, b):
    for i in range(len(b)):
        rotated_b = b[i:] + b[:i]
        if rotated_b in a:
            return True
     return False
```

What would be the return value of `cycpattern_check("xyzw","xyw")`?

CoT Analysis by GPT-4:
The function is designed to check if the second word “b” or any of its rotations is a substring of the first word a. The word b is "xyw". The function will rotate “b” through each possible position and checks if the rotated version is a substring of “a”. The possible rotations of b ("xyw") are:
"xyw" (original)
"ywx" (rotate left by 1)
"wxy" (rotate left by 2)
It will then check if any of these rotations exist as a substring in a ("xyzw").
Checking each rotation against a:
"xyw" is a substring of "xyzw". The function returns True. → Incorrect

While the model correctly explain code statements and also enumerate all rotations, it incorrectly identifies “xyw” is a subset of “xyzw.”