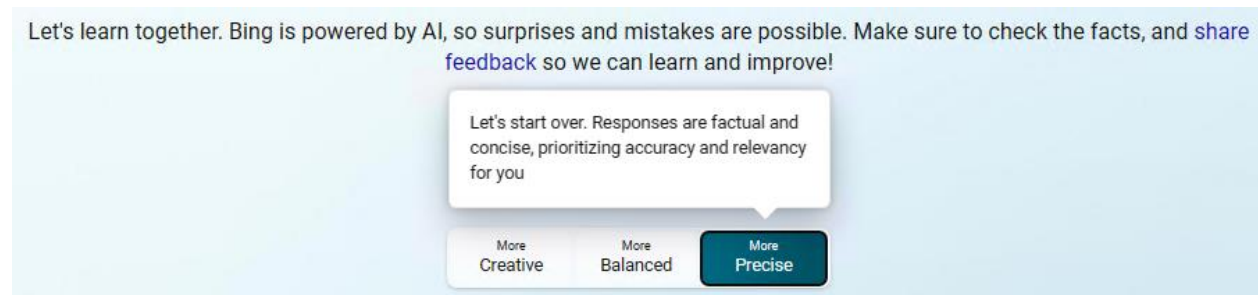


## GPT4 programming understanding is impressive

OpenAI's GPT4 model has just been officially unveiled yesterday and in the [OpenAI GPT4 paper](#) we can see that GPT4 model excels on many diverse academic and professional exams, e.g. Law, Medicine, History, Economics, Math, Chemistry, Physics, Computer Science, etc. On the OpenAI benchmarks, GPT4 performed as well as a top human specialist ( $\geq$ percentile 90<sup>th</sup>).

In his recent article [The End of Programming ? Not Really!](#), Professor Luis Caires had great examples where ChatGPT - which is based on OpenAI GPT 3.5 model – failed badly while reasoning on relatively simple but tricky programming problems.

Luis Caires problems are a good mini benchmark for GPT4. And we can test GPT4 on Bing already. For this test I used the 'More Precise' mode of Bing which lowers the temperature of the GPT4 model to make it less likely to hallucinate:



Here are my interactions with Bing, with the verbatim problems from [Luis Caires article](#) as the prompt:

Implement an infinite precision bit counter with a linked list where each node stores 0 or 1. the counter supports an increment operation. Please use python

Here is an example implementation of an infinite precision bit counter using a linked list in Python:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class BitCounter:
    def __init__(self):
        self.head = Node(0)

    def increment(self):
        current = self.head
        while current:
            if current.data == 0:
                current.data = 1
                return
            else:
                current.data = 0
                if not current.next:
                    current.next = Node(0)
                current = current.next

    def __str__(self):
        result = ''
        current = self.head
        while current:
            result += str(current.data)
            current = current.next
        return result[::-1]

counter = BitCounter()
for i in range(10):
    counter.increment()
    print(counter)
```

This code defines a `Node` class to represent each node in the linked list and a `BitCounter` class to represent the counter. The `BitCounter` class has an `increment` method that increments the value of the counter by one. The method works by traversing the linked list from the head and flipping bits from 0 to 1 until it finds a bit that is already 1. When it finds a bit that is already 1, it flips it to 0 and continues to the next bit. If it reaches the end of the list without finding a bit that is already 1, it appends a new node with value 0 to the end of the list.

Is there anything else you would like to know?

3 de 15 ●



Can you explain how this code works?

How can I modify this code to support decrement operation?

How can I use this code in my program?

If we paste Bing's code in Python and execute we see the output is:

```
1
10
11
100
101
110
111
1000
```

1001  
1010

Which is correct. Each line above is the binary representation of the numbers from 1 to 10.

The second problem is:

Is there any value  $K > 0$  for which the following code will not print "Hi!". `{ i = K; while (i > 1) { if (i % 2 == 0) { i = i / 2 } else { i = 3 * i + 1; } print ("Hi!"); }`

The code you provided is an implementation of the Collatz conjecture. The conjecture states that for any positive integer  $K$ , the sequence generated by the iterative process in the code will eventually reach the number 1. When  $i$  reaches 1, the condition for the `while` loop becomes false and the loop terminates. At this point, "Hi!" is printed.

Although the Collatz conjecture has not been proven yet, it has been tested extensively for a wide range of values and no counterexample (a value of  $K$  for which the sequence does not reach 1) has been found. So it is widely believed that there is no value of  $K > 0$  for which this code will not print "Hi!".

Is there anything else you would like to know?

4 de 15 ●

[What is Collatz conjecture?](#) [Has anyone ever tried to prove it?](#) [How can I learn more about this topic?](#)

This is a perfect answer. Bing/GPT4 recognizes this is the Collatz conjecture and correctly asserts that, assuming it's true, there is no value of  $K > 0$  for which the code won't print Hi.

for which argument values does this function terminate? `int f(int x) { if (x > 0) then return 1+f(x+1); else return 1 }`

The function `f` you provided is a recursive function. It calls itself with an argument of `x+1` if `x > 0`. Since `x+1` is always greater than `x`, this means that the function will keep calling itself with larger and larger values of `x` and will never reach the base case where `x <= 0`. As a result, the function will not terminate for any value of `x > 0`.

For any value of `x <= 0`, the function will immediately return 1 without making any further recursive calls. So the function will terminate for any value of `x <= 0`.

In summary, the function terminates if and only if its argument `x` is less than or equal to 0.

Is there anything else you would like to know?

5 de 15 ●

[What is a recursive function?](#) [How can I fix this code so it terminates for all values of x?](#) [Can you give me an example of a terminating recursive function?](#)

Great answer too and note how helpful Bing's follow up questions are.

The next problem is a bit trickier and I had to slightly update the prompt to clarify the statements and add ; at the end of each. Without the clarification GPT4 would not attempt to re-order and would evaluate the statements in the provided order and say, the result is  $y = -3$  (which is correct, but the goal is to re-order to get to 6). With the slightly rephrased question we have:

Given that initially  $x=y=t=0$ , write the code statements below, using each only once, in such an order so that the final value of  $y = 6$ . The statements are:  $t = x + y$ ;  $x = -1$ ;  $y = t * 3$ ;  $y = t + 3$ ;

Here is one possible order of the code statements that would result in the final value of  $y$  being 6:

```
x = -1
t = x + y
y = t + 3
y = t * 3
```

Is there anything else you would like to know?

1 de 15 ●



What is the final value of  $x$ ?

What is the final value of  $t$ ?

Can you explain how this works?

This is correct.

Let's play a simple number game. We always start from the result 0. Now I let's fix some rules. Each rule has a name and allows you to add some number to the current result. For example, rule "apple" adds 2, and rule "pear" adds 5. So "apple" "pear" gives 7. Using these rules, tell me please a sequence of rule names that gives 9.

Sure! Here's one possible sequence of rule names that gives 9: "pear" "apple" "apple". This would result in  $0 + 5 + 2 + 2 = 9$ .

Is there anything else you would like to know?

2 de 15 ●



Can you give me another sequence?

What if we add a new rule?

Can you explain how this works?

This is a correct as well, but not as impressive as the previous problems were much trickier.

Finally, another problem Caires also posed in a post that GPT 3.5 failed miserably and GPT4 got right:

how many times will the `printf` statement run for each single iteration of the outer for loop ? `for(i=0; i<=2; i++) { int j = 2*i; while (j<4*i){ printf("foo"); j++; } }`

The number of times the `printf` statement will run for each single iteration of the outer `for` loop depends on the value of `i`. For each value of `i`, the inner `while` loop will run until the condition `j < 4*i` is no longer true. Since `j` is initialized to `2*i` and incremented by 1 in each iteration of the inner loop, this means that for each value of `i`, the inner loop will run  $(4*i) - (2*i)$  times. So for each iteration of the outer loop:

- When  $i=0$ : The inner while loop condition is false from start so it won't run at all.
- When  $i=1$ : The inner while loop will run  $(4*1) - (2*1) = 2$  times.
- When  $i=2$ : The inner while loop will run  $(4*2) - (2*2) = 4$  times.

So in total, for a single iteration of the outer for loop, `printf` statement will run 0 time when  $i=0$ ; 2 times when  $i=1$ ; and 4 times when  $i=2$ .

Is there anything else you would like to know?

1 de 15 ●

■ Pare de responder

**This is truly impressive. I did not expect that GPT-4 would be able to solve all these problems correctly.** It's surprising a language model can do this kind of reasoning. GPT4 is remarkable progress on top of GPT3 in such a short amount of time.

Apropos, using GPT4 on Bing rather than on OpenAI's site has two main advantages:

- Bing answer is not stale, it retrieves fresh data from the web as context to the LLM and references the relevant web pages (was not needed here because no scraping was needed to answer these problems)
- Bing automatically suggests good follow up questions to better understand a topic. This is very helpful and is visible in the examples above.

Another momentaneous advantage is that, for now, GPT4 is restricted to paying users of ChatGPT while Bing is free (though some users will go into a waitlist).