

Rebug 1:

First I used binary ninja to provide a decompilation of the program; I also renamed a couple of symbols to make it a bit easier to read.

```
int32_t main(int32_t argc, char** argv, char** envp)

printf("Enter the String: ")
void buffer
__isoc99_scanf(&data_2017, &buffer)
int32_t i = 0
while (*(&buffer + i) != 0)
    i = i + 1
if (i != 0xc)
    printf("that isn't correct, im sorry!")
else
    puts("that's correct!")
    int64_t rax_5 = EVP_MD_CTX_new()
    EVP_DigestInit_ex(rax_5, EVP_md5(), 0)
    EVP_DigestUpdate(rax_5, &data_202a, 2, &data_202a)
    int32_t var_41c = 0x10
    void var_418
    EVP_DigestFinal_ex(rax_5, &var_418, &var_41c, &var_418)
    EVP_MD_CTX_free(rax_5)
    void var_448
    for (int32_t i_1 = 0; i_1 <= 0xf; i_1 = i_1 + 1)
        sprintf(&var_448 + i_1 * 2, "%02x", *(&var_418 + i_1))
    printf("csawctf{%s}\n", &var_448)
return 0
```

Then I put part of the binary into ChatGPT and asked for a description.

ChatGPT Input:



```
printf("Enter the String: ")
void buffer
__isoc99_scanf(&data_2017, &buffer)
int32_t i = 0
while (*(&buffer + i) != 0)
    i = i + 1
if (i != 0xc)
    printf("that isn't correct, im sorry!")
else
    puts("that's correct!")
```

describe this snippet of code



## Output”



This code seems to be written in C or a similar programming language. Let's break it down:



1. `printf("Enter the String: ")`: This line prompts the user to enter a string by displaying the message "Enter the String: ".
2. `void buffer`: This line seems to be a syntax error. It appears to be an attempt to declare a variable named `buffer` without specifying its type. It should be something like `char buffer[MAX_SIZE];`, where `MAX_SIZE` is the maximum size of the buffer.
3. `__isoc99_scanf(&data_2017, &buffer)`: It appears that this line is trying to read input from the user into the `buffer` variable. The function `scanf` or a similar function might have been intended here. It should have the format specifier indicating the type of input expected and should pass the address of `buffer` for input.
4. `int32_t i = 0`: This line declares and initializes an integer variable `i` to zero.
5. `while (*(&buffer + i) != 0) i = i + 1`: This loop seems to be trying to count the number of characters in the `buffer` string until it finds a null terminator (ASCII value 0), which marks the end of a string.
6. `if (i != 0xc) printf("that isn't correct, im sorry!") else puts("that's correct!")`: Here, if the number of characters counted in the string is not equal to 12 (assuming `0xc` is representing 12 in hexadecimal), it will print "that isn't correct, I'm sorry!" Otherwise, it will print "that's correct!".

However, the code has several issues and lacks clarity. It seems to be missing necessary variable declarations, has a syntax error with `void buffer`, uses an unusual method to count string length, and might have a logical error in the comparison (assuming `0xc` is supposed to represent the expected length in hexadecimal).

It seems that the code is intended to receive a string, count its length, and verify if it matches a specific length (12 characters in this case). But it's important to correct syntax errors, define the `buffer` properly, and use appropriate functions to achieve the desired functionality.

From here, I asked it to give me a flag that worked and it tried to do its best.



give me an example string that matches this condition



Certainly! Considering the code expects a string length of 12 characters (assuming `0xc` represents 12 in hexadecimal), here's an example of a string that meets this condition:



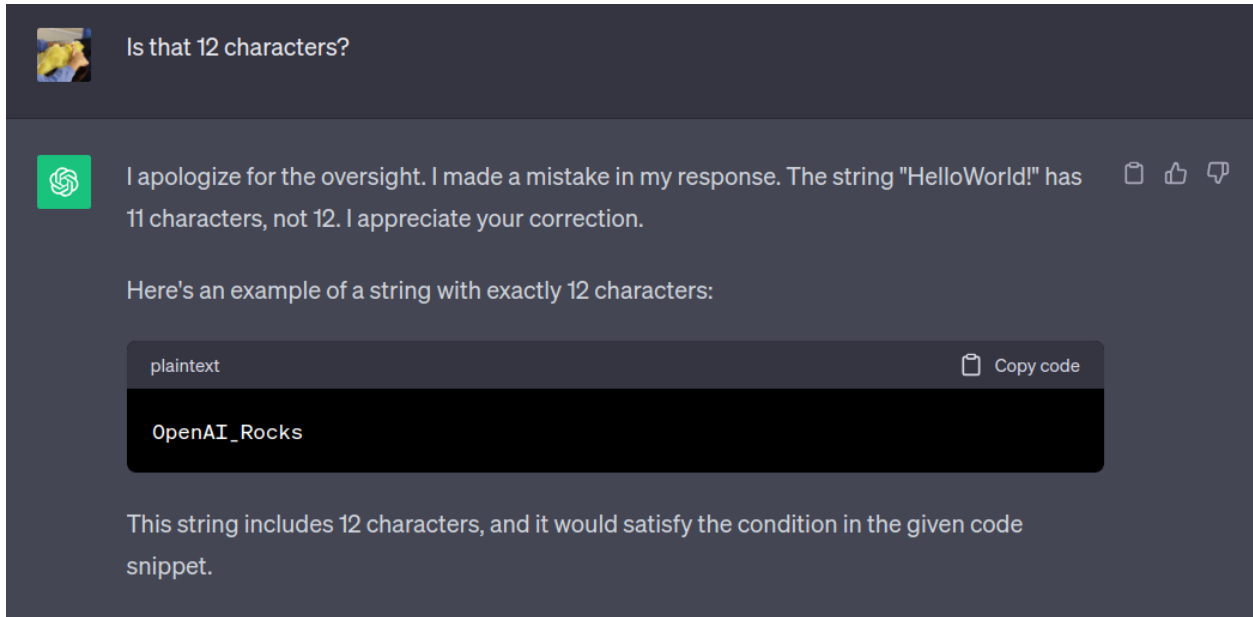
plaintext

Copy code

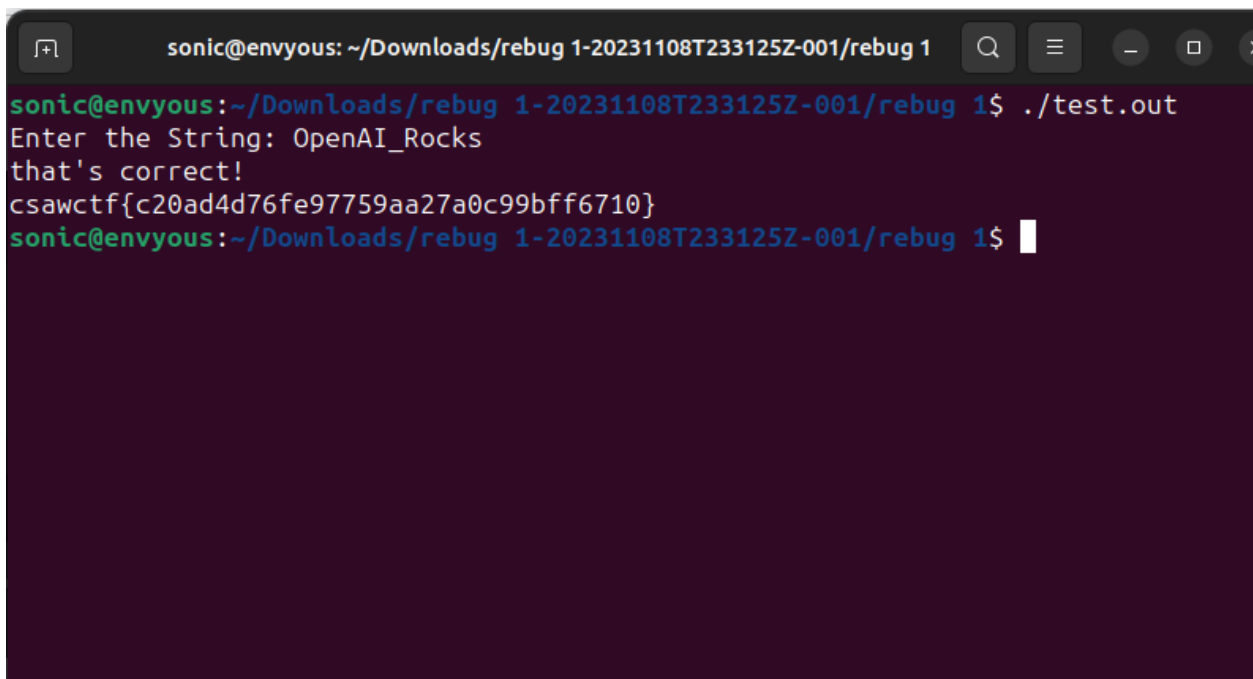
```
HelloWorld!
```

This string contains exactly 12 characters, including the exclamation mark, which means it would satisfy the condition set in the code.

HelloWorld! Is not 12 characters so I asked it again and this time it gave a valid string.



From here, I put this string into the challenge and received the flag as a result



Flag: csawctf{c20ad4d76fe97759aa27a0c99bff6710}0