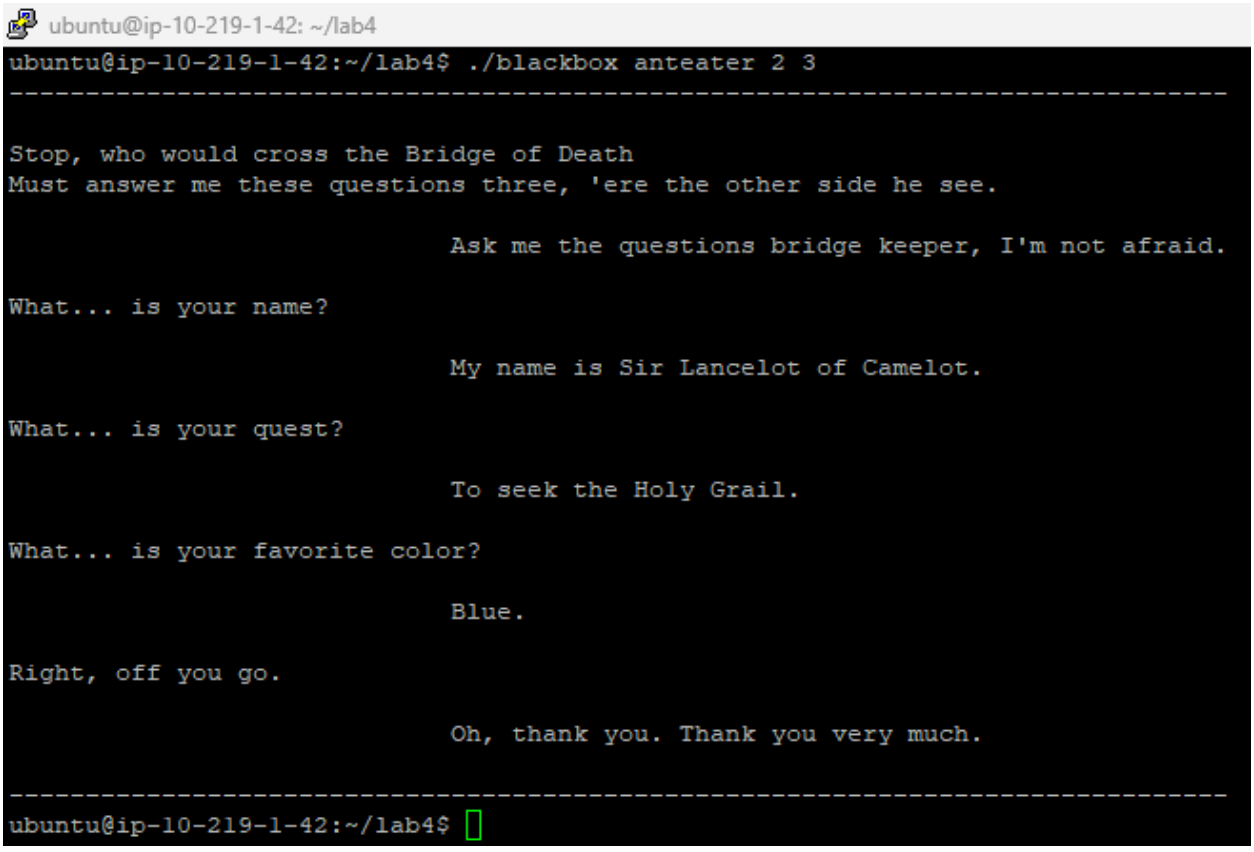
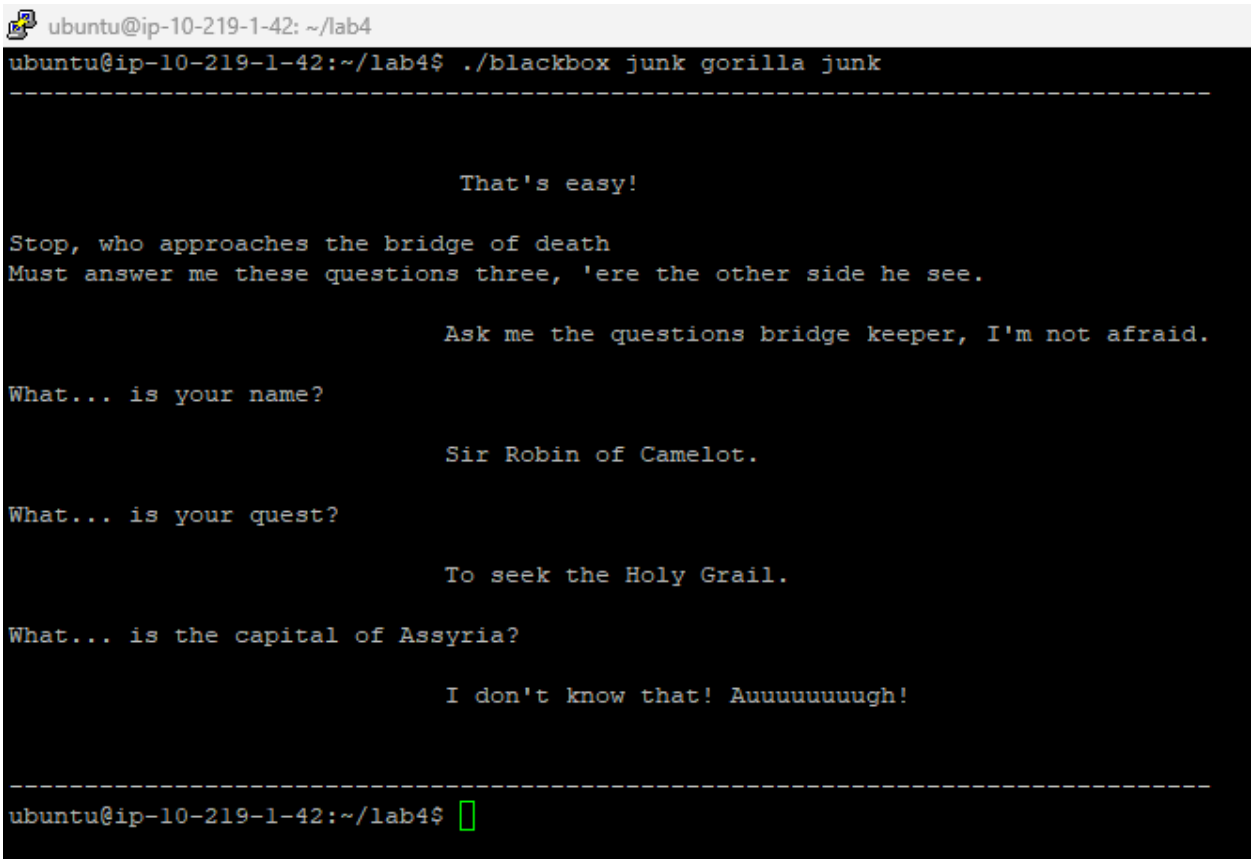
Matthew Adams

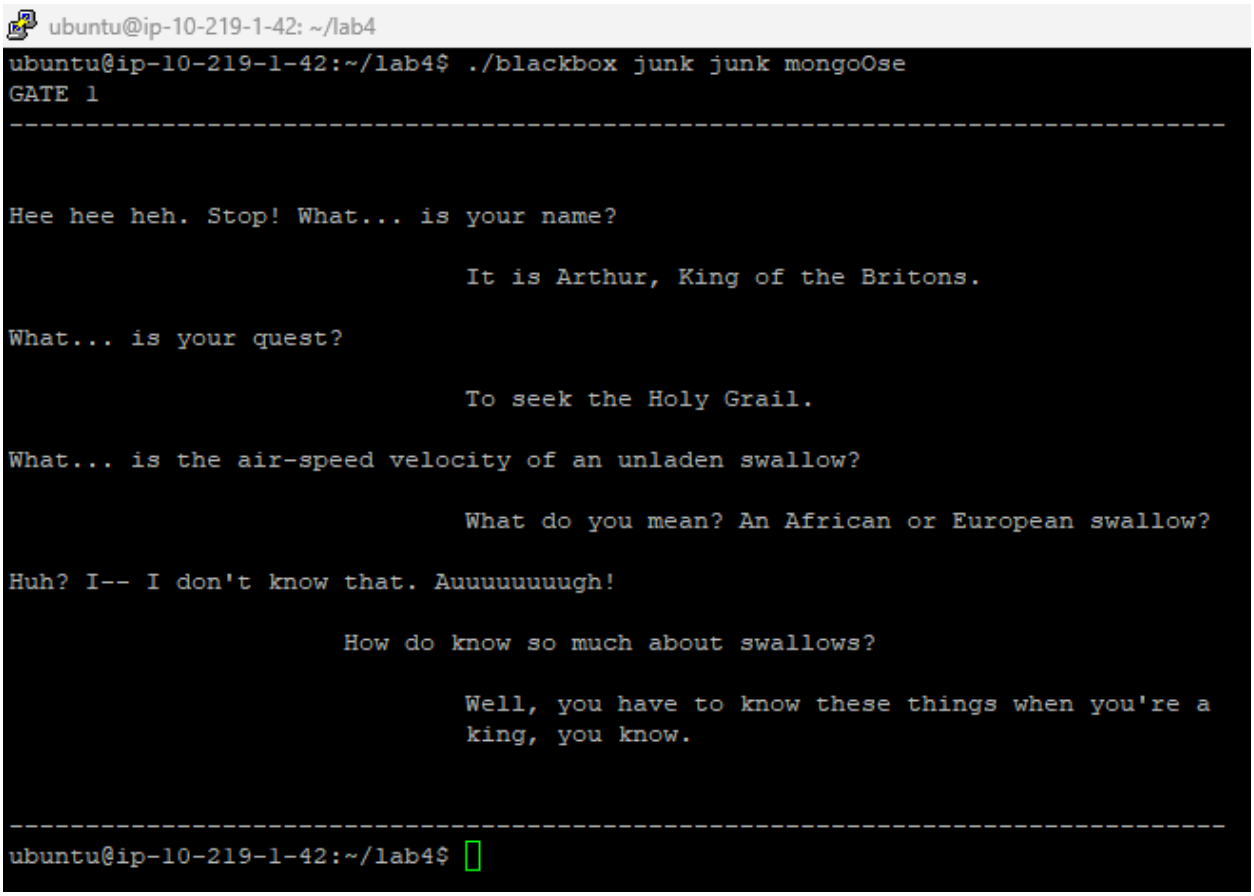
- **Flag 1**
  - Flag 1 Looks very simple. If argc is greater than three and argv[1] is anteater then it calls the flag 1 function.
    - The full command to print this is ./blackbox anteater 2 3



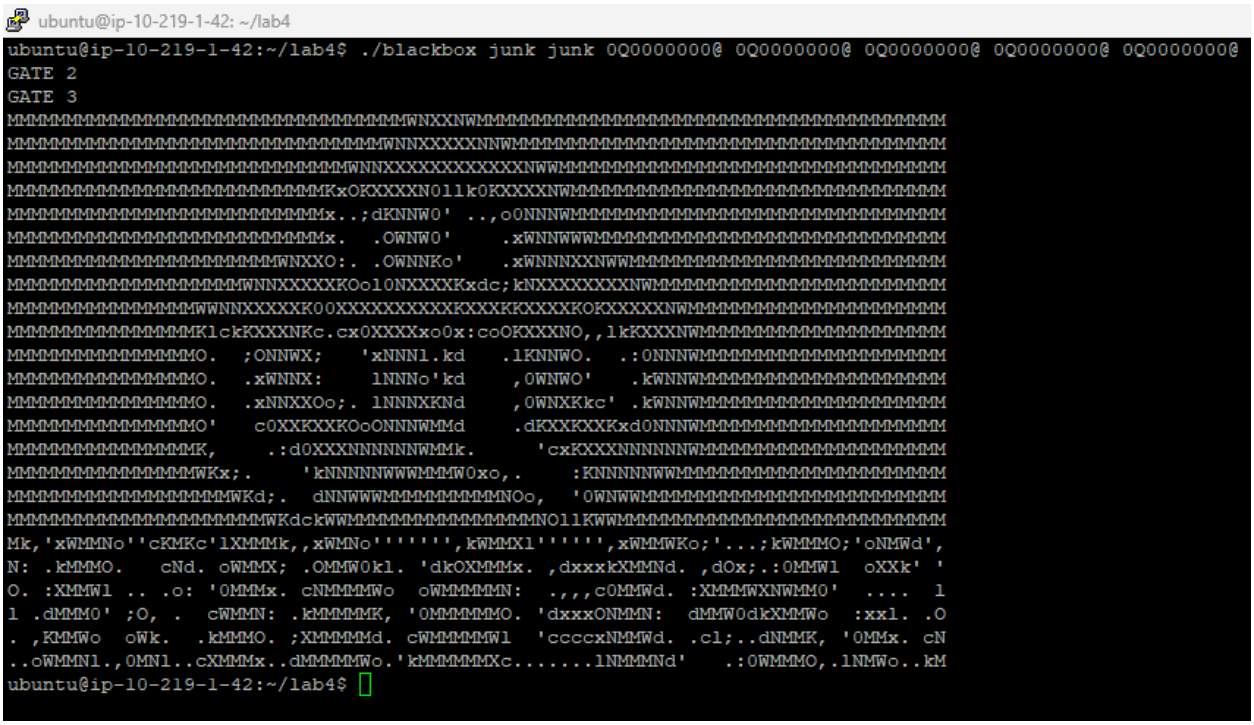
- **Flag 2**
  - Flag 2 Needs similar conditions to flag 1. First there needs to be more than two args passed in. Then there is a for loop that calls the process_args function on every argument. Inside the process_args function, if the passed in string is gorilla it calls the flag 2 function.
    - The full command to print this and previous flags is ./blackbox anteater gorilla 3



- **Flag 3**
  - Flag 3 is a little bit harder. It starts the same as flag two. First there needs to be more than two args passed in. Then there is a for loop that calls the process_args function on every argument. Inside the process_args function, if the arg passed in is mongoose then the pay_attention function is called with the parameters 1, and the argument. Inside the pay attention function, we have counter that tells how many times a gate has been passed through, then it calls the gate function. In this case our gate function is 1 and the arg value being passed in is mongoose. The gate function is checking to see if when the 4 and 5 letters are cast as ints and subtracted it needs to equal 32. To do this it needs to be spelled mongoOse.
    - The full command to print this and previous flags is ./blackbox anteater gorilla mongoOse



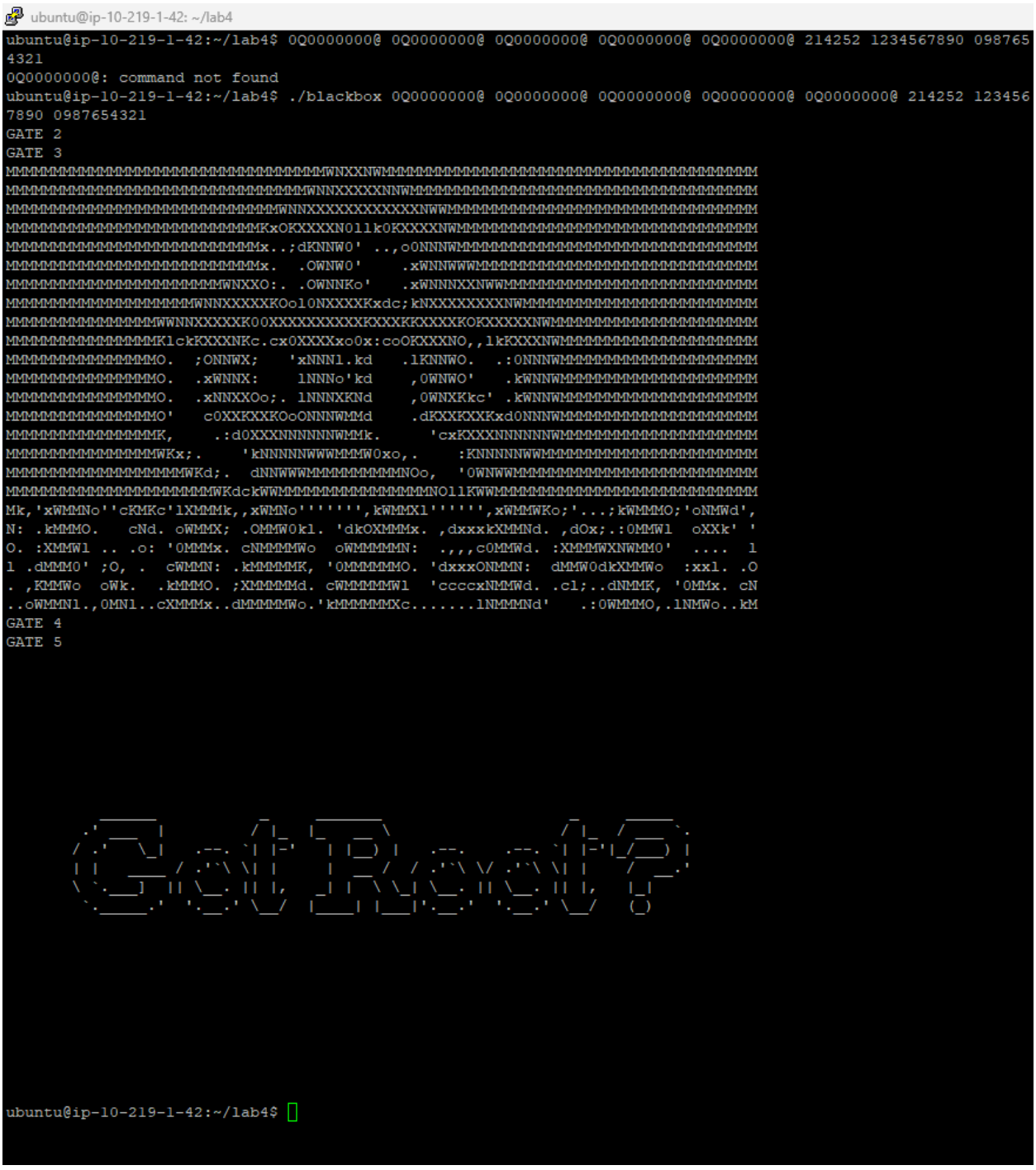
- **Flag 4**
  - Flag four was interesting. First there needs to be more than two args passed in. Then there is a for loop that calls the process_args function on every argument. Inside the process_args function if the [9] item in in the passed in arg string is @ then the gate 2 function is called. In gate 2, gate 2 needs to be passed through 4 times, then gate 3 is called. In gate 3 the [1] char in the string need to be Q. It looks something like this 0Q0000000@
    - The full command to print this and previous flags is ./blackbox anteater gorilla mongoOse 0Q0000000@ 0Q0000000@ 0Q0000000@ 0Q0000000@ 0Q0000000@



- **Flag 5**
  - Flag 5 is like flag 4. First there needs to be more than two args passed in. Then there is a for loop that calls the process_args function on every argument. Inside the process_args function if the 1st and 2nd characters subtracted to be 1 and the 3rd and 4th characters subtracted to be 2, and the 5th and 6th characters subtracted to be 3 then it would call pay_ attention passing in gate 4 and the string provided for the math in the If statement. Gate 4 required that gate 1 be passed through at least 1 time.
    - The full command to print this and previous flags is: ./blackbox anteater gorilla mongoOse 0Q0000000@ 0Q0000000@ 0Q0000000@ 0Q0000000@ 0Q0000000@ 214252

```
ubuntu@ip-10-219-1-42:~/lab4$ ./blackbox junk mongoOse 214252
GATE 1
--------------------------------------------------------------------------------


Hee hee heh. Stop! What... is your name?

                              It is Arthur, King of the Britons.

What... is your quest?

                              To seek the Holy Grail.

What... is the air-speed velocity of an unladen swallow?

                              What do you mean? An African or European swallow?

Huh? I-- I don't know that. Auuuuuuuugh!

                      How do know so much about swallows?

                              Well, you have to know these things when you're a
                              king, you know.


--------------------------------------------------------------------------------
GATE 4
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWWMWWWMMMMMMMMMMWWWMMMWWMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWNXXNWMNOkk0NMMMWNWMMMWWWWMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMWWWKkolx0NW0c.  .oKWWXOxx0NWMMWWMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMWMMNkc..l0NWWk'     ;0WWW0l',o0NMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMWMMWWNk;. 'xNMXdlc.      ,lcxNWNx. .cOWMWMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMWWMXl.  ,OWWWk. .          ;0WWWx.  .oXMWWMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMWWWK:   .kWWWMW0c.        .oKWMWWWo.   :KWWMWWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMWMKc    lNWWWWMMWO,      ;OWWWWMMWK,    :KMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMWd.    dMMMMMMMMWO.    ,0MMMMWWMWNc    .dWMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMX:     oWMWWMWWMWNc    lWMMWWMMMMX:     :XMMWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMWMMK,     ,0WMMMMMMWK;    :KWMMMMMMWk'     ,0MWWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMWWMK,      :0WMMMMWKl.    .lKWMMMMNk,      ,0MWWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMX:       .cxk0xl'        .cxkko;.       ;XMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMWWx.                                    .dWMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMXc                                    ;0WWWMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMWMMMWM0,                                  .kWWWWWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMW0:                                ,kWMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMWWXd.                            .lKWWWMMWWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMWMMMWWMW0l.                        .cOWMMMWMMWWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMWKx:.                 ..:xKWWMMMMWMMWMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMWXkdc;'.......',:ldOXWMMMWMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMWNNXXXKXXXNNNWMMMWWMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMWWMMMMMWWWWMMMMWWMWWWWWWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
ubuntu@ip-10-219-1-42:~/lab4$
```

- **Flag6**
    - Flag 6 is also in process_args. First there needs to be more than two args passed in. Then there is a for loop that calls the process_args function on every argument. Inside the process_args there is a for loop that compares the previous argument to the next one in revers order, if they are mirrored and the don't have repeating characters then count will equal 10. When count equals 10 pay_attention is called, passing in 5 and argv_val. This calls gate 5. To get gate 5 to print the flag gates 3 and 4 must have been gone through one time, and gate 1 gone through0 times.
        - the command should look something like this ./blackbox 0Q0000000@ 0Q0000000@ 0Q0000000@ 0Q0000000@ 0Q0000000@ 214252 123456 7890 0987654321



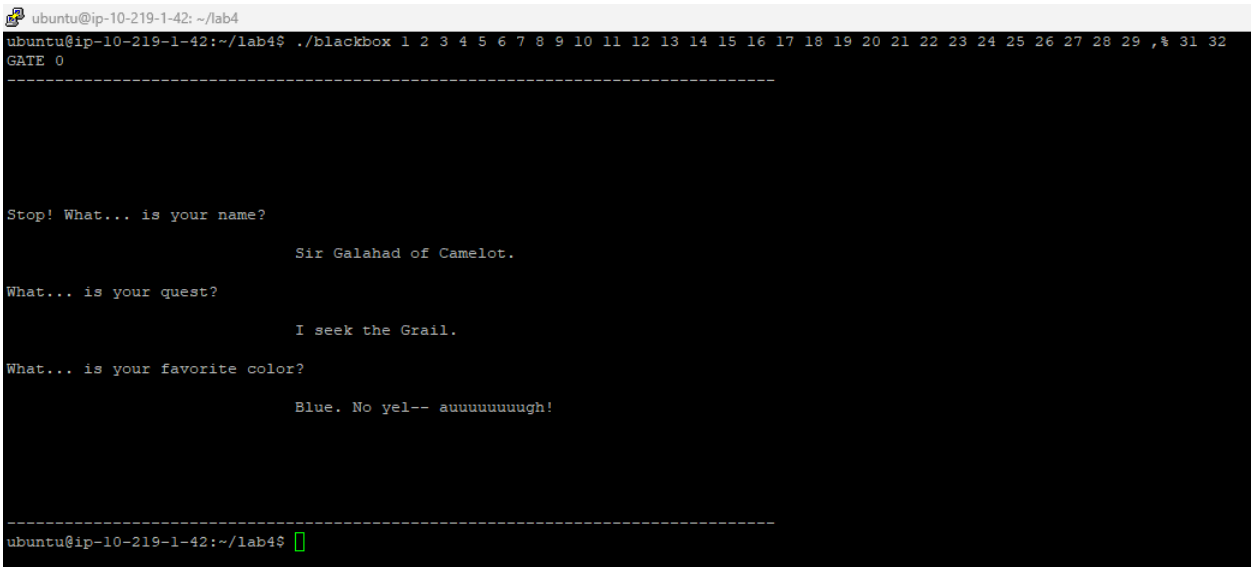
- **Flag7**
  - I was stuck on this flag for awhile. This flag was also in process_args. In process_args there is and if that is checking arg_val[0] is equal to the number of times gate 1 has been gone through, and if arg_val[1] is equal to the number of times gate 4 has been gone through. After that you can get to gate 6, to get

through gate 6 the xor of arg_val[2] and arg_val[3] needs to be 32, also known as the same number as a capitol, then lower case

- The command I used looks like this: ./blackbox gorilla mongoose 214252 $'\x01\x01'Aa


ubuntu@ip-10-219-1-42: ~/lab4
ubuntu@ip-10-219-1-42:~/lab4$ ./blackbox gorilla mongoose 214252 $'\x01\x01'Aa
GATE 1
GATE 4
GATE 6
ubuntu@ip-10-219-1-42:~/lab4$

- **Flag8**
  - Flag 8 was easier. In main there is an if to see if 33 arguments were passed in. If t 33 args where passed in pay_attention is call with the parameters of 0 and the 30th value passed in. This will call gate 0. Gate 0 checks to se of the param passed in is equal to ,%. If it is it prints the flag
    - Command is ./blackbox 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 ,% 31 32



- **Flag9**
  - Flag 9 was a bit different. Flag 9 is in process envp. In main there is a loop that goes through every item in the evnp char** array. Process_envp is called in this loop with the values of the current envp value and the first argument passed in. Inside process_envp there is an if comparing if the value for envp is equal to the string deadbeef. If it is equal then payattention is called with the vals 7 and the argument passed in. This will call gate 7. Gate 7 checks to see if the arg passed in is 11 characters long. If it is it then the flag is printed.
    - The command is deadbeef="" ./blackbox 11111111111

- **Flag10**
  - Flag 10 is also in process envp. Just like in flag 9 there is a loop that calls every envp var with the first passed in value in argv. In process envp there is and if else statement. To call flag 9 you want to get into the if statement, for flag ten you want the else statement. Inside the else statement there is a function called. I named it getTime. Inside this getTIme function the local system time is saved and returned in a varable. This variable is then returned and compared to our passed in envp value. Once the two are equal flag ten is printed. I wrote a c function that basically copied this code and gave me the date its comparing, I then tried converting it into a byte string and I was not getting it to print.