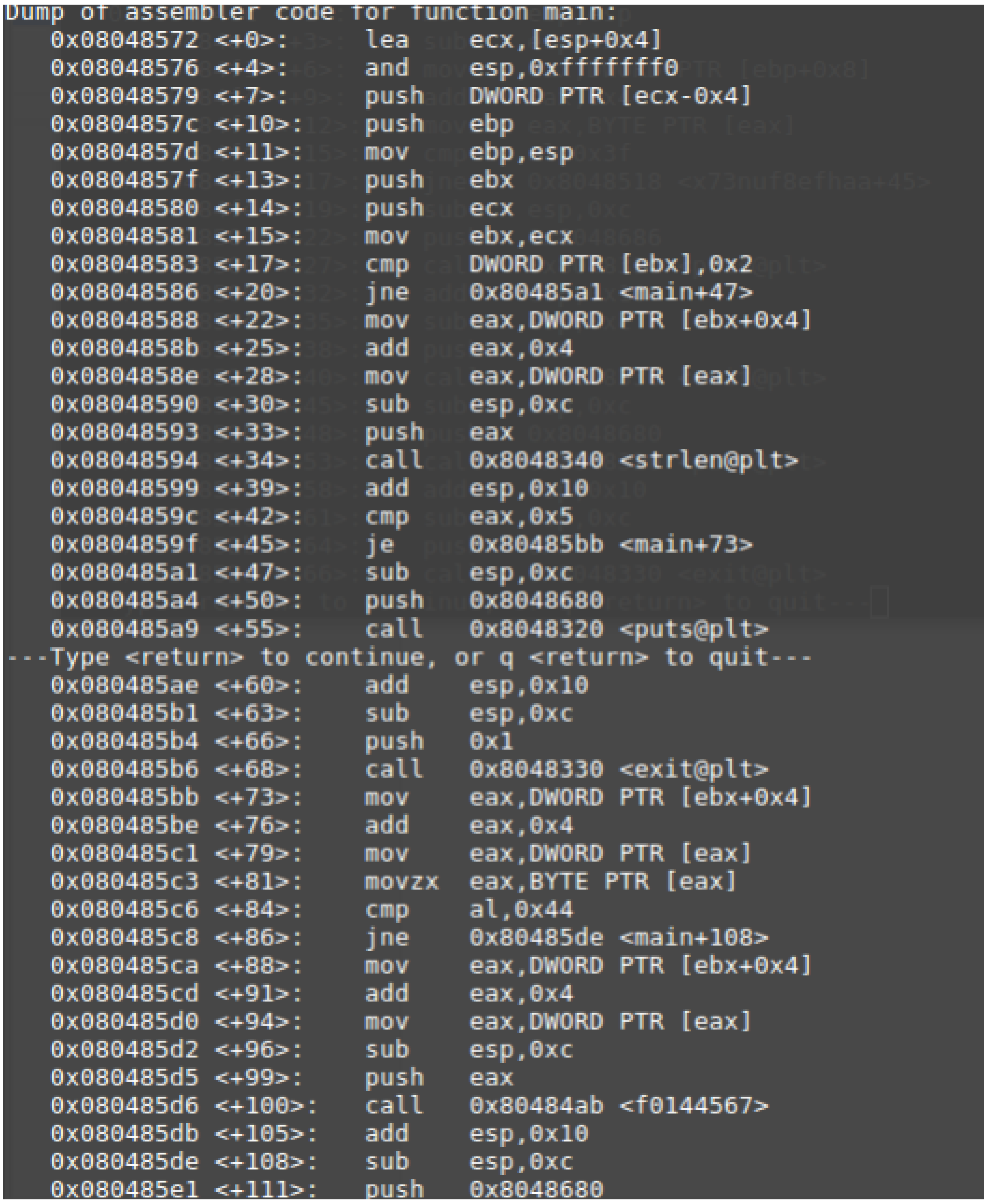
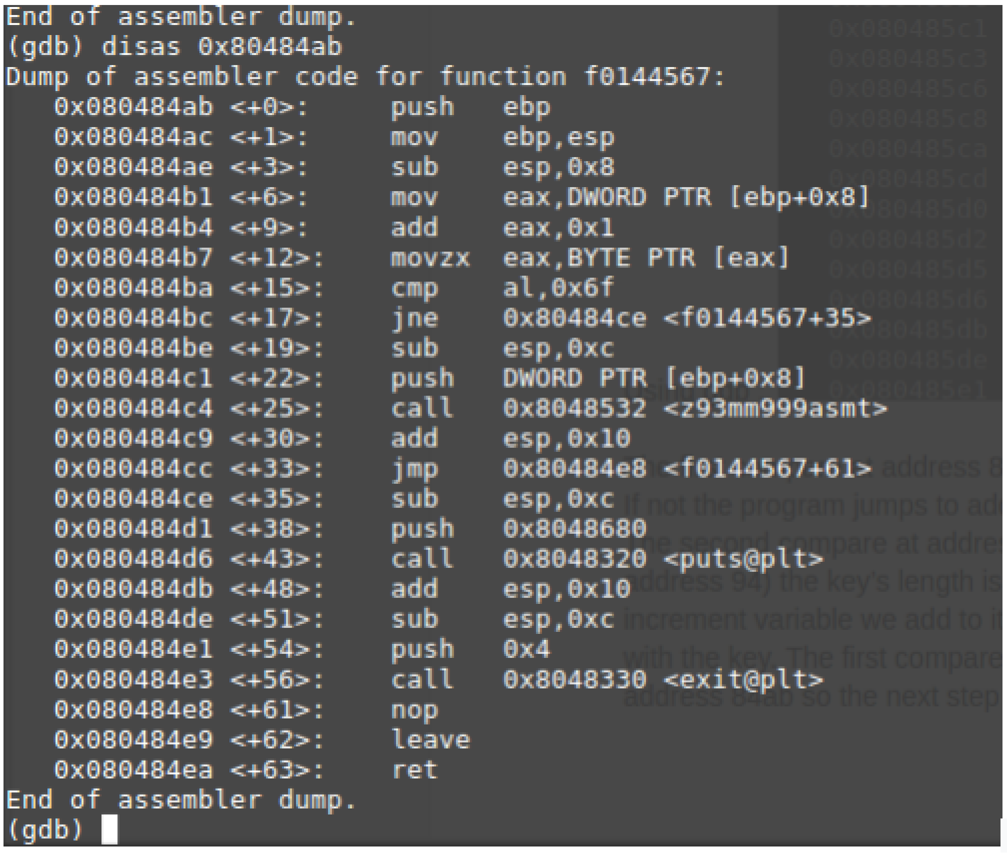
I used GDB in order to solve this challenge.



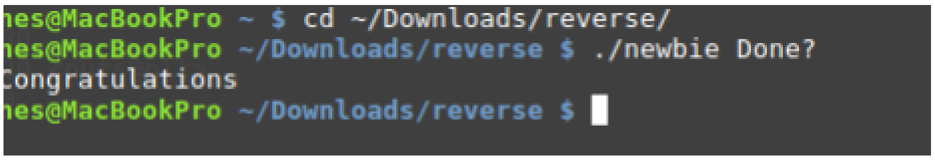
```
Dump of assembler code for function main:
   0x08048572 <+0>:	lea    ecx,[esp+0x4]
   0x08048576 <+4>:	and    esp,0xfffffff0
   0x08048579 <+7>:	push   DWORD PTR [ecx-0x4]
   0x0804857c <+10>:	push   ebp
   0x0804857d <+11>:	mov    ebp,esp
   0x0804857f <+13>:	push   ebx
   0x08048580 <+14>:	push   ecx
   0x08048581 <+15>:	mov    ebx,ecx
   0x08048583 <+17>:	cmp    DWORD PTR [ebx],0x2
   0x08048586 <+20>:	jne    0x80485a1 <main+47>
   0x08048588 <+22>:	mov    eax,DWORD PTR [ebx+0x4]
   0x0804858b <+25>:	add    eax,0x4
   0x0804858e <+28>:	mov    eax,DWORD PTR [eax]
   0x08048590 <+30>:	sub    esp,0xc
   0x08048593 <+33>:	push   eax
   0x08048594 <+34>:	call   0x8048340 <strlen@plt>
   0x08048599 <+39>:	add    esp,0x10
   0x0804859c <+42>:	cmp    eax,0x5
   0x0804859f <+45>:	je     0x80485bb <main+73>
   0x080485a1 <+47>:	sub    esp,0xc
   0x080485a4 <+50>:	push   0x8048680
   0x080485a9 <+55>:	call   0x8048320 <puts@plt>
---Type <return> to continue, or q <return> to quit---
   0x080485ae <+60>:	add    esp,0x10
   0x080485b1 <+63>:	sub    esp,0xc
   0x080485b4 <+66>:	push   0x1
   0x080485b6 <+68>:	call   0x8048330 <exit@plt>
   0x080485bb <+73>:	mov    eax,DWORD PTR [ebx+0x4]
   0x080485be <+76>:	add    eax,0x4
   0x080485c1 <+79>:	mov    eax,DWORD PTR [eax]
   0x080485c3 <+81>:	movzx  eax,BYTE PTR [eax]
   0x080485c6 <+84>:	cmp    al,0x44
   0x080485c8 <+86>:	jne    0x80485de <main+108>
   0x080485ca <+88>:	mov    eax,DWORD PTR [ebx+0x4]
   0x080485cd <+91>:	add    eax,0x4
   0x080485d0 <+94>:	mov    eax,DWORD PTR [eax]
   0x080485d2 <+96>:	sub    esp,0xc
   0x080485d5 <+99>:	push   eax
   0x080485d6 <+100>:	call   0x80484ab <f0144567>
   0x080485db <+105>:	add    esp,0x10
   0x080485de <+108>:	sub    esp,0xc
   0x080485e1 <+111>:	push   0x8048680
```

The first compare at address 0x08048583 is to determine the right number of arguments which naturally has to be 2. If not the program jumps to address 0x080485a1 and prints wrong via the puts' call. The second compare at address 0x0804859c is to compare the length of the key (The program calls strlen before at address 0x08048594) the key's length is 5. If it's the case it jumps to address 0x080485bb. The program is comparing character by character the string entered with the key. The first compare is with the value 0x44 which corresponds to D in ASCII. At address 0x080485d6 it calls a function at address 0x080484ab so the next step is to look at this address to determine the remaining characters of the key



```
End of assembler dump.
(gdb) disas 0x80484ab
Dump of assembler code for function f0144567:
   0x080484ab <+0>:     push   ebp
   0x080484ac <+1>:     mov    ebp,esp
   0x080484ae <+3>:     sub    esp,0x8
   0x080484b1 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x080484b4 <+9>:     add    eax,0x1
   0x080484b7 <+12>:    movzx  eax,BYTE PTR [eax]
   0x080484ba <+15>:    cmp    al,0x6f
   0x080484bc <+17>:    jne    0x80484ce <f0144567+35>
   0x080484be <+19>:    sub    esp,0xc
   0x080484c1 <+22>:    push   DWORD PTR [ebp+0x8]
   0x080484c4 <+25>:    call   0x8048532 <z93mm999asmt>
   0x080484c9 <+30>:    add    esp,0x10
   0x080484cc <+33>:    jmp    0x80484e8 <f0144567+61>
   0x080484ce <+35>:    sub    esp,0xc
   0x080484d1 <+38>:    push   0x8048680
   0x080484d6 <+43>:    call   0x8048320 <puts@plt>
   0x080484db <+48>:    add    esp,0x10
   0x080484de <+51>:    sub    esp,0xc
   0x080484e1 <+54>:    push   0x4
   0x080484e3 <+56>:    call   0x8048330 <exit@plt>
   0x080484e8 <+61>:    nop
   0x080484e9 <+62>:    leave
   0x080484ea <+63>:    ret
End of assembler dump.
(gdb)
```

The second compare is at address 0x080484ba. 0x6f is o in ASCII. At address 0x080484c4 the program calls another function at 0x8048532. After looking at it we can determine the third letter and following the same method (ie each time jumping to the address of the call) I determined the following characters of the key.



```
nes@MacBookPro ~ $ cd ~/Downloads/reverse/
nes@MacBookPro ~/Downloads/reverse $ ./newbie Done?
Congratulations
nes@MacBookPro ~/Downloads/reverse $
```