	HW10	- -	.0/8/5/1/0	COLLABORA	ATOR(S):
			kind of attack d against?	oes a stack	guard or stack canary
					annary value have and
3/1/0			an explanation f	or each of t	those values:
3/1/0) b)			
3/1/0) с)			
	strc	ру()		a stash sma	ence below, for each ash would be detected and
3/1.	/0 a	.)			
3/1.	^{/0} b				<pre>int main() { char buf[8]; strcpy(buf, "Go Navy"); //(a) strcpy(buf, "Go Navy!"); //(b)</pre>
3/1,	/0 C)			strcpy(buf, "Beat Army"); //(c) strcpy(buf, "Beat Army!"); //(d)
3/1,	^{/0} d	.)			

__/25

N A ME:	

4. Ignoring **gcc** additions for stack cannaries, write the equivalent C code for the function foo:

```
Dump of assembler code for function foo:
           0x0804854d <+0>: push
                                                                                                                      ebp
           0 \times 0804854e <+1>:
                                                                                      mov
                                                                                                                      ebp, esp
           0 \times 08048550 <+3>:
                                                                                      sub esp.0x28

        0x08048553
        <+6>:
        mov
        eax,DWORD PTR [ebp+0x8]

        0x08048556
        <+9>:
        mov
        DWORD PTR [ebp-0x1c],eax

        0x08048559
        <+12>:
        mov
        eax,gs:0x14

        0x0804855f
        <+18>:
        mov
        DWORD PTR [ebp-0xc],eax

        0x08048562
        <+21>:
        xor
        eax,eax

        0x08048564
        <+23>:
        lea
        eax,[ebp-0x11]

        0x08048567
        <+26>:
        mov
        DWORD PTR [ebp-0x18],eax

        0x0804856a
        <+29>:
        nop

        0x0804856b
        <+30>:
        mov
        eax,DWORD PTR [ebp-0x18]

        0x08048571
        <+36>:
        mov
        DWORD PTR [ebp-0x18],edx

        0x08048574
        <+39>:
        mov
        edx,DWORD PTR [ebp-0x1c]

        0x08048573
        <+42>:
        lea
        ecx,[edx+0x1]

        0x08048570
        <+48>:
        mov
        DWORD PTR [ebp-0x1c],ecx

        0x08048580
        <+51>:
        mov
        BYTE PTR [eax]

        0x08048582
        <+53>:
        movzx
        eax,BYTE PTR [eax]

        0x08048585
        <+56>:
        test
        al,al

           0x08048553 <+6>:
                                                                                     mov eax, DWORD PTR [ebp+0x8]
           0x08048585 <+56>: test al,al
0x08048587 <+58>: jne 0x804
                                                                                                                      0x804856b <foo+30>
           0x08048589 <+60>: nop
           0x0804858a <+61>: mov eax,DWORD PTR [ebp-0xc]
          0x0804858d <+64>: xor eax, DWORD PTR gs:0x14

0x08048594 <+71>: je 0x804859b <foo+78>

0x0804859b <+73>: call 0x8048340 <__stack_ch
                                                                                                                      0x8048340 < stack chk fail@plt>
           0 \times 0804859c < +79>:
                                                                                     ret
```

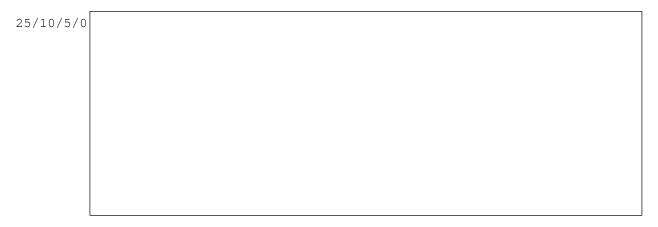
25/10/5/0



/25 **2 of 3**

HAME	

5. Using the same dissasembly from the previous question, add in the equivalent C code based on the gcc additions for stack cannaries.



5/3/1/0 6. Are stack cannaries the **same or different** for each function call within a single process? Explain.



5/3/1/0 7. Are stack cannaries the **same or different** for children of the processes that **do not call exec()**? Explain.



5/3/1/0 8. Are stack cannaries the **same or different** for chidlren of the process that **do call exec()**? Explain.



5/3/1/0 9. Is it possible to circumvent stack cannaries in GDB? If so, explain the process, if not, explain why not.



 $^{5/3/1/0}$ 10. Explain the challenges associated with **brute forcing** a stack cannary? How many guesses would it take to have a 25% chance of getting the cannary right at least once.

