CST 334 (Operating Systems)

Dr. Glenn Bruns

# Lab: Computer Architecture

The purpose of this lab is to get a better understanding of the stack and the heap. Work with a partner, and refer to the hints at the end if you get stuck.

1. Get stack.c from this directory on mlc104: /home/CLASSES/brunsglenn/cst334/labs/comp-arch-lab/stack.c

If you can't access mlc104, use this [stack.c](https://drive.google.com/open?id=1p1tcDgWanHhy9pRjNy2j8iTqwyZXigq6_J6ohkPG6w4).

Read the code carefully, starting with main(). Take your time. Predict what the printf statements will output.

1. Copy it to your own directory and and try compiling it. You should get some warning messages.
2. Fix the warning message you get about ‘printf’.
3. Fix the warning message you get about ‘malloc’.
4. Explain the warning message about a local variable. What is the problem you are being warned about?
5. Try running the program. Does it do what you expect?
6. Comment out the line p2 = foo(1,4), and compile and run the code again. Does it do what you expect?
7. Variable yp is a local variable of bar(), just like y is a local variable of foo(). So why doesn't bar() suffer from the same problem as foo()?
8. Explain what is happening, and how it relates to the warning message about a local variable.
9. What is the lesson you've learned about writing C code?

**Hints**:

2. To copy it, use command 'cp'. To compile: gcc -o stack stack.c

3. hint 1: implicit declaration means the function wasn’t declared before it was used. This usually indicates an #include is missing. hint 2: to find which include, try command ‘man 3 printf’ at the command line. See the SYNOPSIS section.

4. See the hints for 3.

5. hint 1: If a local variable stored in the stack, the heap, or somewhere else? hint 2: when a function returns, what happens to the memory used to store its local variables?

6. Did you expect the first print statement to print "foo(1,3) = 4"?

7. Actually, there is a difference between the two cases. In foo(), a pointer to a local variable is returned. In bar(), the value of the local variable itself is returned. Why does this make a difference? Hint: is there any problem in actually returning the value from a function?

8. In a call to foo(), the returned value is an address of a local variable. This is an address in the stack region of memory. When the second foo() call is made, that part of the stack region is reused. In the first call to bar(), an address to somewhere in the heap region is returned. That part of the heap region is not reused when the second call to bar(). This is tricky and requires some thought.

9. Don't return an address in the stack region!