

Name: Jack Stawasz

Course: CSCI 312 Principles of Programming Languages

Assignment Deadline: April 9, 2025

Question 1 (Play Around with Array/Pointer Arguments)

Make a new directory called Assignment4 in your ppl repo. Make a new directory called Assignment4/Question1 that will contain your source code and executables for this question. Complete *Play Around with Array/Pointer Arguments* (Expert C Programming p. 249):

1. Implement 1 in a function called one that main calls in a file called play.c. Make ca local to main. Record your answer to 1 here:

0x7fff9947ef28

0x7fff9947ef56

0x7fff9947ef57

2. Implement 2 in a function called two that main calls in play.c. Make pa local to main. Record your answer to 2 here:

0x7fff9947ef28

(nil)

0x1

0x1

3. Implement 3 in main (which calls one(ga) and two(ga)) in play.c. Record your answer to 3 here:

one(ga) output:

0x7fff9947ef28

0x55e1d2b4b010

0x55e1d2b4b011

0x55e1d2b4b011

two(ga) output:

0x7fff9947ef28

0x55e1d2b4b010

0x55e1d2b4b011

0x55e1d2b4b011

COMPARISON: the outputs are identical. The local variable form of **ga** could be stored in different locations between function calls **one()** and **two()**, but due to stack memory optimization the address is the same because once one function is done with the space the other function claims it. Function **two** also prints **++ga** while **one** does not, but this expression is identical to **&(ga[1])** as they both display the address of the byte after the pointer.

4. Implement 4 in main in play.c. Record your answer to 4 here:

0x55e1d2b4b010

0x55e1d2b4b010

0x55e1d2b4b011

5. Record your answer to 5 here:

EXPECTATION: I expect all the outputs of `one()` to be unique for both inputs `ca` and `ga` because the address of the array is a pointer stored separately from the array contents and the array contents for indices 0 and 1 will both have defined memory locations. The `two()` function, however, will have identical values for `&(pa[1])` and `++pa` because these two forms have the same meaning.

OUTCOME: the results of each problem are shown in the above parts (1-4). My expectations were all correct except for two things. First, I should have mentioned that `&(pa[0])` would print `(nil)` because that space in memory is reserved for the value `pa` points to, but since the variable was only defined and not initialized there is no value there. Second, the printed value of `&ga` is different when called in a function locally vs when called in `main()` because when `ga` is passed into a function the new local variable is a pointer to `ga`.

Continue to use branching to get more practice.