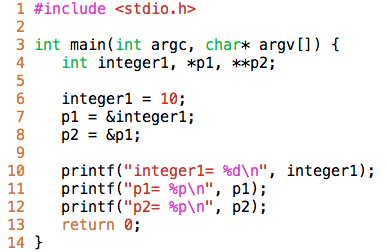
# CpSc 1011 Lab 10

# Pointers – Answer Sheet

### Your Name: Joseph Laudati

**Warm Up**

1. Compile and run this program:



2. Symbol Table:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Address** |
| integer1 | int | 0x7fff8c1ff994 |
| p1 | int \* | 0x7fff8c1ff988 |
| p2 | int \*\* | 0x7fff8c1ff980 |

Memory Chunk (address on the left, data on the right):

|  |  |
| --- | --- |
| 0x7fff8c1ff994 | 10 |
| 0x7fff8c1ff988 | 10 |
| 0x7fff8c1ff980 | 10 |

* 1. (12 pts – each box is 4 pts) Fill in the above memory chunk table to reflect the changes that lines 6, 7, and 8 cause.
  2. (4 pts) In we substitute lines 10-12 with the following two statements, then what will be the output of the program?

(\*p1)++;

printf(“integer1= %d\n”, \*p1);

**Results:**

integer1= 10

integer1= 11

p1= 0x7ff7b1db634c

p2= 0x7ff7b1db6340

* 1. (4 pts) Will the output be the same if we substitute the above two lines with the following two statements?

integer1++;

printf(“integer1= %d\n”, \*p1);

**Results:**

integer1= 10

integer1= 11

p1= 0x7ff7b26ec34c

p2= 0x7ff7b26ec340

* 1. (4 pts) Will the output be the same if we substitute the above two lines with the following two statements?

\*p1++;

printf(“integer1= %d\n”, \*p1);

**Result:**

integer1= 10

integer1= -1316882920

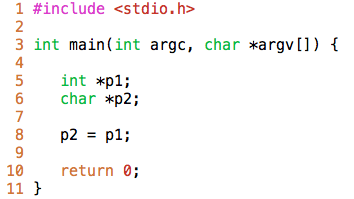
p1= 0x7ff7b181f350

p2= 0x7ff7b181f340

* 1. (6 pts) Explain why the above outputs are the same or different from each other.

*They are different because the program is changing the values for each variable. Integer prints its initial value and is then changed as the program tells it to.*

1. (12 pts) Consider the following program and answer the questions below.



What does variable p2 represent? Will this program successfully compile without warnings? Why or why not? (Try it!)

*P2 represents p1 in character form. The program will not compile because the there are incompatible pointer types that are causing conflict with the output. Because p1 is an integer and p2 is a character, they will cause an argument called “Mismatch error”*

**Arrays and Pointers**

OUTPUT:

0

1

2

3

4

5

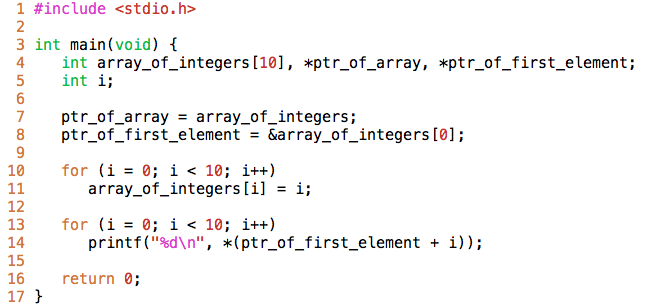
6

7

8

9

4. Compile and run the following program and answer the corresponding questions.



* 1. (4 pts) What is the output of the program (use the red box to the right)?
  2. (6 pts) What does array\_of\_integers evaluate to? If you dereference it, what value do you get? Try it.

Array\_of\_integers evaluate to an Array of 10 integers. When you dereference it, you will receive the number 0 because it is the first value you start with.

* 1. Suppose we change lines 10-11 to the following:

for (i = 0;i < 10; i++)

\*(ptr\_of\_array + i) = i;

(6 pts) Will the output change? Why or why not?

No the output does not change because the pointer to the first element is already referenced to the first element.

* 1. Suppose we change lines 13-14 of the original program to the following statements:

for (i = 0;i < 10; i++)

printf(“%d\n”, \*(ptr\_of\_array + i));

(6 pts) Will the output change? Why or why not?

The output will change because you are now telling the program to reprint the same array on top of the previous array.

* 1. Suppose we change lines 10-11 of the original program to the following statements:

for (i = 0;i < 10; i++)

\*(ptr\_of\_array++) = i;

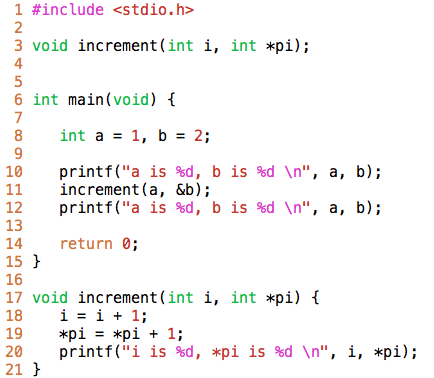
(6 pts) Will the output change? Why or why not?

**No** the output will not change because it is basically taking in the same information like the original code and following the same instruction that tells it to print the array. The only difference being that the information being used is being called from a different variable. The different variable takes information from the original source.

(6 pts) Now, does the variable ptr\_of\_array have the same value as the variable ptr\_of\_first\_element? Why or why not?

**No, they are different BUT they point to the same memory location.** This is the reason why the outputs are identical while two pieces of code are different.

**Function and Pointers**

1. a. (12 pts) What is the output of the following code (use the red box below)?

b. (12 pts) Why? (i.e. **show what’s going on in memory**)

It is doing this because a is stored in one location while B is stored in another. When the function is called, the function increments variable “A” by one while it does the same later for B. Then the code prints these variables for the user.

OUTPUT:

a is 262487776, \*pi is -1337264800

i is 2, \*pi is -1337265352

a is 1, \*pi is 3