

CPSC 1011 Lab 10 - Pointers

Elise Bloom

TOTAL POINTS

88 / 100

QUESTION 1

Question 2a 12 pts

1.1 Part 1 4 / 4

✓ **+ 4 pts** Answer is ``10``.

+ **0 pts** Answer is not ``10``.

1.2 Part 2 4 / 4

✓ **+ 4 pts** Answer is ``0x7fff8c1ff994``.

+ **0 pts** Answer is not ``0x7fff8c1ff994``.

1.3 Part 3 4 / 4

✓ **+ 4 pts** Answer is ``0x7fff8c1ff988``.

+ **0 pts** Answer is not ``0x7fff8c1ff988``.

QUESTION 2

2 Question 2b 4 / 4

✓ **+ 4 pts** Answer is **exactly** ``integer1= 11`` (with varying white space).

+ **2 pts** Answer includes ``11``, but is not **exactly** ``integer1= 11`` (with varying white space).

+ **0 pts** Answer does not include ``11``.

QUESTION 3

3 Question 2c 4 / 4

✓ **+ 4 pts** Answer is *`yes`* (or something that otherwise confirms that the output will remain the same), and may include printed output ``integer1=`

`11`` (with varying white space) or the answer ``11``.

+ **2 pts** Answer is not *`yes`* (or something that otherwise confirms that the output will remain the same), but does include printed output ``integer1= 11`` (with varying white space) or the answer ``11``.

+ **0 pts** Answer is not *`yes`* (or something that otherwise confirms that the output will remain the same), and does not include the printed output ``integer1= 11`` (with varying white space) or the answer ``11``.

QUESTION 4

4 Question 2d 4 / 4

✓ **+ 4 pts** Answer is *`no`* (or something that otherwise states that the output will change), and may include printed program output.

+ **2 pts** Answer is not *`no`* (or something that otherwise states that the output will change), but does include printed program output.

+ **0 pts** Answer is not *`no`* (or something that otherwise states that the output will change), and does not include printed program output.

QUESTION 5

5 Question 2e 6 / 6

✓ **+ 6 pts** Answer is similar to the following:

"part (d) is different because it increments the address of what `p1` is pointing to, and then de-references that other memory location"

+ 0 pts Answer is not similar to the following:

"part (d) is different because it increments the address of what `p1` is pointing to, and then de-references that other memory location"

QUESTION 6

6 Question 3 8 / 12

+ 12 pts Answer includes (and/or is similar to) all of the following:

- * `p2` represents a pointer to a character
- * program will compile with warnings
- * incompatible pointer types

✓ **+ 8 pts** Answer includes (and/or is similar to)

**only* two of three of the following:*

- * `p2` represents a pointer to a character
- * program will compile with warnings
- * incompatible pointer types

+ 4 pts Answer includes (and/or is similar to)

**only* one of three of the following:*

- * `p2` represents a pointer to a character
- * program will compile with warnings
- * incompatible pointer types

+ 0 pts Answer does not include or is not at all similar to the following:

- * `p2` represents a pointer to a character
- * program will compile with warnings
- * incompatible pointer types

QUESTION 7

7 Question 4a 4 / 4

✓ **+ 4 pts** Answer is:

`0`
`1`
`2`
`3`
`4`
`5`
`6`
`7`
`8`
`9`

+ 2 pts Answer is: `0-9` with varying white space.

+ 0 pts Answer is not:

`0`
`1`
`2`
`3`
`4`
`5`
`6`
`7`
`8`
`9`

or any collection of `0-9` with varying white space.

QUESTION 8

8 Question 4b 6 / 6

✓ **+ 6 pts** Answer includes (and/or is similar to) both of the following:

** `array\_of\_integers` evaluates to the location in memory of the beginning of the array, i.e. the first*

element of the array

** if you de-reference it, you get the first value in the array, which is `0`*

+ 3 pts Answer includes (and/or is similar to)

**only* one of the following:*

** `array\_of\_integers` evaluates to the location in memory of the beginning of the array, i.e. the first element of the array*

** if you de-reference it, you get the first value in the array, which is `0`*

+ 0 pts Answer does not include or is not similar to the following:

** `array\_of\_integers` evaluates to the location in memory of the beginning of the array, i.e. the first element of the array*

** if you de-reference it, you get the first value in the array, which is `0`*

QUESTION 9

9 Question 4c 3 / 6

+ 6 pts Answer includes (and/or is similar to) both of the following:

** no*

** initializing same array elements via pointer*

✓ **+ 3 pts** Answer includes (and/or is similar to)

**only* one of the following:*

** no*

** initializing same array elements via pointer*

+ 0 pts Answer does not include or is not similar to the following:

** no*

** initializing same array elements via pointer*

QUESTION 10

10 Question 4d 3 / 6

+ 6 pts Answer includes (and/or is similar to)

both of the following:

** no*

** printing same array elements via pointer*

✓ **+ 3 pts** Answer includes (and/or is similar to)

**only* one of the following:*

** no*

** printing same array elements via pointer*

+ 0 pts Answer does not include or is not similar to the following:

** no*

** printing same array elements via pointer*

QUESTION 11

Question 4e 12 pts

11.1 Part 1 4 / 6

+ 6 pts Answer includes (and/or is similar to) all of the following:

** no, output does not change*

** de-references the pointer, assigns a value, then increments the pointer*

** fills up the array with the correct values in the correct locations*

✓ **+ 4 pts** Answer includes (and/or is similar to)

**only* two of three of the following:*

** no, output does not change*

** de-references the pointer, assigns a value, then increments the pointer*

** fills up the array with the correct values in the correct locations*

+ 2 pts Answer includes (and/or is similar to)

**only* one of three of the following:*

- * no, output does not change
- * de-references the pointer, assigns a value, then increments the pointer
- * fills up the array with the correct values in the correct locations

+ 0 pts Answer does not include or is not at all similar to the following:

- * no, output does not change
- * de-references the pointer, assigns a value, then increments the pointer
- * fills up the array with the correct values in the correct locations

11.2 Part 2 6 / 6

✓ **+ 6 pts** Answer includes (and/or is similar to) both of the following:

- * no
- * the address of where `ptr_of_array` is pointing to after the initialization is now one past the end of the array

+ 3 pts Answer includes (and/or is similar to) only one of the following:

- * no
- * the address of where `ptr_of_array` is pointing to after the initialization is now one past the end of the array

+ 0 pts Answer does not include or is not similar to the following:

- * no
- * the address of where `ptr_of_array` is pointing to after the initialization is now one past the end of the array

QUESTION 12

12 Question 5a 12 / 12

✓ **+ 12 pts** Answer is:

``a is 1, b is 2``

``i is 2, *pi is 3``

``a is 1, b is 3``

+ 6 pts Answer is:

``a is 1, b is 2``

``i is 2, *pi is 3``

``a is 1, b is 3``

with non-exact but similar white space or capitalization/punctuation.

+ 0 pts Answer is not:

``a is 1, b is 2``

``i is 2, *pi is 3``

``a is 1, b is 3``

or any similar output with varying white space or capitalization/punctuation.

QUESTION 13

13 Question 5b 12 / 12

✓ **+ 12 pts** Answer explains or depicts that ``i`` is incremented locally within the function ``increment``, then the pointer ``pi`` is incremented locally within the function ``increment``, which affects the actual value of ``b`` since it was passed as the argument for ``pi``.

****Note:**** Answers for this question may be highly variable. A similar answer is acceptable as long as it explains why each value is incremented at the right

time.

+ 0 pts Answer does not explain or depict that `i` is incremented locally within the function `increment`, then the pointer `pi` is incremented locally within the function `increment`, which affects the actual value of `b` since it was passed as the argument for `pi`.

****Note:**** Answers for this question may be highly variable. A similar answer is acceptable as long as it explains why each value is incremented at the right time.

QUESTION 14

14 Absence Penalty 0 / 0

✓ **- 0 pts** *Student was present in lab when required.*

- 0 pts Student was not present in lab, but provided a reasonable excuse. Responsible TA(s) will provide more information in comments.

- 50 pts Student was not present in lab, and did not provide a reasonable excuse ahead of time. Responsible TA(s) will provide more information in comments.

CpSc 1011 Lab 10

Pointers – Answer Sheet

Your Name: _____ Elise Bloom

Warm Up

1. Compile and run this program:

```
1 #include <stdio.h>
2
3 int main(int argc, char* argv[]) {
4     int integer1, *p1, **p2;
5
6     integer1 = 10;
7     p1 = &integer1;
8     p2 = &p1;
9
10    printf("integer1= %d\n", integer1);
11    printf("p1= %p\n", p1);
12    printf("p2= %p\n", p2);
13    return 0;
14 }
```

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	0x7fff8c1ff994
0x7fff8c1ff980	0x7fff8c1ff988

- a. (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.

- b. (4 pts) If 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);
```

integer1= 11

- c. (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);
```

Yes

- d. (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);
```

No

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

B and C have the same output because both are changing the value stored in integer1. B dereferences the pointer, accessing the value which p1 points to (integer1), and changing it. C directly changes the value of integer1. D is different than B and C because it changes where p1 is pointing to. Thus, p1 is no longer pointing to integer1, so the printf statement is misleading.

3. (12 pts) Consider the following program and answer the questions below.

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[]) {
4
5     int *p1;
6     char *p2;
7
8     p2 = p1;
9
10    return 0;
11 }
```

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

p2 represents the array containing the argv values. It does not compile without warnings because it says that “p2 = p1” is assigning type int * to char *. The pointer types do not match.

Arrays and Pointers

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

```
1 #include <stdio.h>
2
3 int main(void) {
4     int array_of_integers[10], *ptr_of_array, *ptr_of_first_element;
5     int i;
6
7     ptr_of_array = array_of_integers;
8     ptr_of_first_element = &array_of_integers[0];
9
10    for (i = 0; i < 10; i++)
11        array_of_integers[i] = i;
12
13    for (i = 0; i < 10; i++)
14        printf("%d\n", *(ptr_of_first_element + i));
15
16    return 0;
17 }
```

- a. (4 pts) What is the output of the program (use the red box to the right)?

OUTPUT:

0
1
2
3
4
5
6
7
8
9

- b. (6 pts) What does `array_of_integers` evaluate to? If you dereference it, what value do you get? Try it.

`array_of_integers` is the address of the array storing the integers. If you dereference it, it returns 0.

- c. 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 doesn't change because the dereferenced pointer changes the values stored in the array, which is what the original code did.

- d. 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?

No, the output doesn't change because `ptr_of_array` references the address of `array_of_integers`. In the for loop, `(ptr_of_array + i)` is dereferenced and systematically prints the values of the array.

- e. 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 doesn't change because the pointer is being moved through the memory locations to change each value stored in the array as it goes through the for loop.

- f. (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; at the beginning they had the same value, but the location of `ptr_of_array` has been changed.

Function and Pointers

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

```
1 #include <stdio.h>
2
3 void increment(int i, int *pi);
4
5
6 int main(void) {
7     int a = 1, b = 2;
8
9     printf("a is %d, b is %d \n", a, b);
10    increment(a, &b);
11    printf("a is %d, b is %d \n", a, b);
12
13    return 0;
14 }
15
16
17 void increment(int i, int *pi) {
18     i = i + 1;
19     *pi = *pi + 1;
20     printf("i is %d, *pi is %d \n", i, *pi);
21 }
```

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

The function `increment` receives two values: "a" and the pointer for "b". It increases the value of "a" by 1. Using a pointer, it increases the value of the actual variable "b" by 1. Then it prints these new values. However, as the final output demonstrates, the actual value of "a" has not been changed because the function `increment` just received the value of "a", not access to the actual memory location of "a". Since `increment` did receive "b's" memory location, the actual value of "b" has been changed.

OUTPUT:

```
a is 1, b is 2
i is 2, *pi is 3
a is 1, b is 3
```