CSSE 120 – Introduction to Software Development in C	Quiz 5 from Videos: Binky BoxAndPointerDiagrams (pdfs) PointerSyntax UsingPointers…(pdfs) PointersAsParameters

Name: ____ **Section:** ______ **Grade:** ____________

Answer these questions while viewing the assigned videos. ***Not sure of an answer?*** Ask your instructor to explain ***at the beginning of the next class session***. You can then fill in your answer, still for full credit. (But no fair doing that unless you attempted the question first.)

Videos for this quiz: www.rose-hulman.edu/class/csse/binaries/C-Videos/session5/

Video 1: Binky (3 minutes, 11 seconds)

1. One of the following declares and allocates an *integer*, and one declares and allocates a *pointer to an integer*. Circle the one that allocates a pointer to an integer.

   `int alicia;` **`int* deliah;`**

2. Write a statement that declares and allocates a variable that is a pointer to a *float*. (Choose the name of the variable as you wish.)

3. **True** or **False** (circle one): The value of a pointer variable is an *address* (that is, a number that specifies a place in memory).

4. The video shows one way to assign a value to a pointer, using *malloc*. (We will talk more about *malloc* later.) Another way is to use the *address-of* operator **&**, as shown below:

```
int a;
int* p;
p = &a;
```

   True or **False** (circle one): The value of variable **p** after the code to the right executes is the *address* in memory where the variable **a** is stored.

5. Continuing to use the code shown above, after that code executes:

   True or **False** (circle one): The variable **a** is an integer variable.

   True or **False** (circle one): The value of variable **a** is an integer.

   True or **False** (circle one): The value of variable **a** is an indeterminate “garbage” value that depends on what was left behind in memory before the above statements executed.

   True or **False** (circle one): The variable **p** is a pointer-to-an-integer variable.

   True or **False** (circle one): The value of variable **p** is an address where an integer is stored.

   True or **False** (circle one): The value of variable **p** is an indeterminate “garbage” value that depends on what was left behind in memory before the above statements executed.

   True or **False** (circle one): The value of variable **p** is NOT a “garbage” value; however, its value depends on where the compiler/loader put variables into memory.

6. Continuing to use the code shown above (and repeated below, for your convenience):

```
int a;
int* p;
p = &a;
```

   a. The variable **p** is called a ____________________ (hint: begins with a "p", ends with "***ter***")

   b. The variable **a** is **p**'s ________________________ (hint: begins with a "p", ends with "***tee***")

7. Continuing to use the code shown above, suppose we add another statement:

```
*p = 42;
```

   What does this statement make have the value 42: The variable **p**, or **p**'s pointee? (Circle your choice)

8. Consider the following code:

```
int b;
int* pb;
```

   a. What is the value of **b** after the above executes? (State its value, or say "garbage" if its value is some indeterminate number that depends on what was left behind in memory before the above statements executed.)

   b. What is the value of **pb** after the above executes? (State its value, or say "garbage" if its value is some indeterminate number that depends on what was left behind in memory before the above statements executed.)

   c. Keeping in mind your answers to the above, explain briefly why the following statement may cause the program to crash.

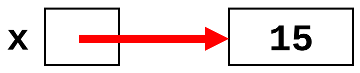
```
int c;
int* pc;
*pc = 13;
```

   d. The above statements might (circle ALL that apply):

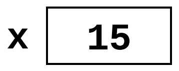
      - Cause the program to crash.
      - Change the value of some other value in the program.
      - Not affect the execution of the program in any apparent way.

Video 2 (actually a set of PDFs): Pointers (including Box-and-Pointer diagrams)

9. **True** or **False** (circle one): In ***Python***, EVERY variable is a *reference* to a value, and hence every variable in Python is depicted in a box-and-pointer diagram like this (with an ***arrow*** going out of the box):



10. **True** or **False** (circle one): In **C**, every ORDINARY variable (that is, every NON-pointer variable) is the *name* for a place in memory, and hence every ordinary (non-pointer) variable in C is depicted in a box-and-pointer diagram like this (with its value shown INSIDE the box):

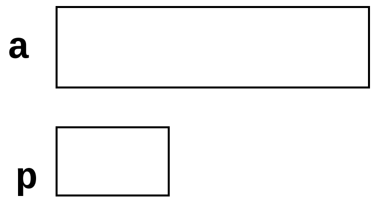


11. Write the three lines of code that accomplish the following:
    - Declare a variable that is a *float*.
    - Declare a variable that is a *pointer* to a float.
    - Set the value of your *pointer-to-float* variable to be the *address* of your *float* variable.

12. Continuing the previous problem, add a 4th line of code that sets the value of your float variable to 12.8 ***without mentioning your float variable***. (Instead, use your pointer variable.)

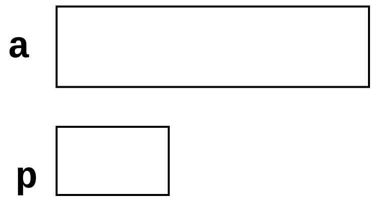
13. Fill in the given box-and-pointer diagram (shown to the right) to indicate what the snippet of code (also shown to the right) does.

```
int main() {
    int a;
    int* p;
    a = 77;
    p = &a;
}
```



14. Fill in the given box-and-pointer diagram (shown to the right) to indicate what the snippet of code (also shown to the right) does.

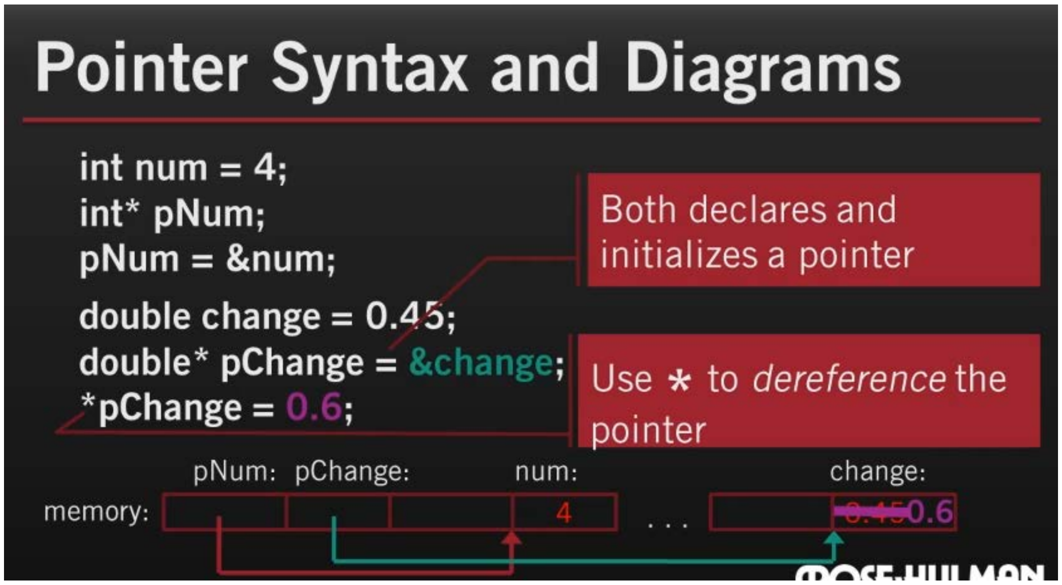
```
int main() {
    int a;
    int* p;
    a = 77;
    p = &a;
    *p = 88;
}
```



Video 3: Pointer Syntax (the concept and notation for pointers) (4 minutes, 33 seconds)

The following questions refer to this screenshot from the video.

After the statements shown have executed:



Pointer Syntax and Diagrams

```
int num = 4;
int* pNum;
pNum = &num;
double change = 0.45;
double* pChange = &change;
*pChange = 0.6;
```

Both declares and initializes a pointer

Use * to *dereference* the pointer

15. **True** or **False** (circle one): The variable **pNum** is a ***pointer*** variable.
16. **True** or **False** (circle one): The value of variable **pNum** is an ***address***.
17. **True** or **False** (circle one): The value of variable **pChange** is the ***address*** of variable **change**.
18. **True** or **False** (circle one): The value of variable **change** is 0.45 after the 4th line above executes, but 0.60 after the 6th (last) line above executes.

19. The symbol that is used in a declaration to indicate that a variable is a pointer variable is ____________.

20. The symbol that is used in an expression to "dereference" a pointer (i.e., to get the pointer's pointee) is ______________.

21. The symbol that is used in an expression to get the address of a variable is _______________.

22. What questions, if any, do you have about the picture shown above? (It is important that you understand every detail of this excellent example.)

Video 4 (actually a set of PDFs): Using Pointers to send information back from a function (using *mutation*)

No questions on this reading, although the reading is critical for exercises you will do in class.

Video 5: Pointers As Parameters (for mutation) (8 minutes, 25 seconds)

No questions on this video, although the video will help you in exercises you will do in class.

If you have other questions about these videos /reading, ask them at your next class session!