**Q.1 [line 55] What is the difference between a struct and a class?**

- The default access for members of a struct are public by default

- The default access for members of a class are private by default

- When deriving a struct, the default access specifier is public

- When deriving a class, the default access specifier is private

**Q.2 [line 63] What are function declarations?**

It tells the compiler about the function's name, return type and parameters. Basically how to call that function.

This is necessary when you \*\*define\*\* a function in a source file, then you call that function in another file.

**Q.3 [line 67] Why are variable names not needed here?**

- A parameter name is required when a compiler needs to know what variable to read/write into.

- A function declaration alone is just a "blueprint" of how a function should work, it's not a working function, so the compiler doesn't need to know parameter names since it just need to know how to create the function when it needs to call it in a different file, not how to work it.

**Q.4 [line 75] Does your IDE know if this method is used?**

If you press Shift + F12, it’ll show where this method is being used in the current document. The location of searching can be modified (currently set to “Entire Solution”).



There would be a green underline under it if it’s not being used.



**Q.5 [line 86] un-initialised values ... what this show and why?**

It doesn’t show anything . . . It’s probably the settings of the IDE that suppressed this, since it happens for me in C# in the same IDE.



**Q.6 [line 95] Did this work as expected?**

Yes. We didn’t give “a” a value. So it didn’t work, as expected.



**Q.7 [line 97] Initialisation list - do you know what they are?**

**Q.8 [line 113] Should show age=1, x=1, y=2. Does it?**

It does.

**Q.9 [line 117] Something odd here. What and why?**

unsigned int means it can’t be negative, yet p1 has -1 as age



****

**Q.10 [line 128] showParticle(p1) doesn't show 5,6,7 ... Why?**

You’re not taking the references of p (by using ->), but you’re taking p’s values (by using . ) So if you pass p1 into it, it would look like 1 = 5, 1 = 6 and 1 = 7, not p1’s age = 5, etc.



**Q.11 [line 153] So what does -> mean (in words)?**

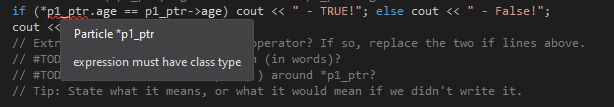
Eg. ptr -> a

You’ll be using the memory address (ptr) to pass the “properties” of that variable (a), not passing its “properties’ values” like Q10

(This helped me find words to use for explanation) <https://www.tutorialspoint.com/What-is-arrow-operator-in-Cplusplus>

**Q.12 [line 154] Do we need to put ( ) around \*p1\_ptr?**

It’s like when you do 5 \* (1 + 2) and you get 15. It’s for indicating precedence. Without it, you’re essentially saying “get the memory address of this value”, which doesn’t make sense . . . like you can’t get a memory address of number 6.



(Answer found here) <http://www.cplusplus.com/forum/beginner/138279/>

**Q.13 [line 160] What is the dereferenced pointer (from the example above)?**

p1

**Q.14 [line 165] Is p1 stored on the heap or stack?**

**Q.15 [line 166] What is p1\_ptr pointing to now? (Has it changed?)**

No it hasn’t changed.

Before (memory address is affc18)



After (same memory address, but with different values)



**Q.16 [line 172] Is the current value of p1\_ptr good or bad? Explain**

**Q.17 [line 175] Is p1 still available? Explain.**

**Q.18 [line 180] <deleted - ignore> :)**

**Q.19 [line 189] Uncomment the next code line - will it compile?**

**Q.20 [line 192] Does your IDE tell you of any issues? If so, how?**

**Q.21 [line 200] MAGIC NUMBER?! What is it? Is it bad? Explain!**

**Q.22 [line 207] Explain in your own words how the array size is calculated.**

**Q.23 [line 375] What is the difference between this function signature and**

**Q.24 [line 380] Uncomment the following. It gives different values to those we saw before**

**Q.25 [line 219] Change the size argument to 10 (or similar). What happens?**

**Q.26 [line 237] What is "hex" and what does it do? (url in your notes)**

**Q.27 [line 242] What is new and what did it do?**

**Q.28 [line 252] What is delete and what did it do?**

**Q.29 [line 256] What happens when we try this? Explain.**

**Q.30 [line 265] So, what is the difference between NULL and nullptr and 0?**

**Q.31 [line 267] What happens if you try this? (A zero address now, so ...)**

**Q.32 [line 302] Are default pointer values in an array safe? Explain.**

**Q.33 [line 317] We should always have "delete" to match each "new".**

**Q.34 [line 325] Should we set pointers to nullptr? Why?**

**Q.35 [line 330] How do you create an array with new and set the size?**