Read the standards: Undefined, Unspecified and implementation-defined behavior in C

With abundant information comes abundant mis-information. This holds true for the internet and I cannot emphasize more the perils of allowing anyone and everyone to be an expert on the internet.

While being a C language trainer at an institute that specializes in professional courses, I would often stumble upon some particularly frustrating questions that some of the students walked up to me with for clarifications. Some of those came from "reputed" websites that provided practice quizzes for interview prep, and most of these websites never seemed to put in more than half an effort into the quality of such questions. But surprisingly, more came from evaluation tests from the hiring organizations themselves.

No question is stupid, except when it is.

Take for example, this oft-recurring MCQ that I have lately been seeing everywhere, and I sincerely hope it doesn't make way into my nightmares.

```
#include <stdio.h>

int main() {

    int num= 5;

    num= num++ + ++num + ++num + num++; // i aM VeRy sMaRT

    printf("Bet you didn't expect %d\n", num);

    return 0;

}
```

Very cool sir. No, I am not going to pick up my paper and pen trying to evaluate that.

Hope you had fun iterating over the values of num?

"You see, the way the post-decrement operator works is ….." **STOP!**

You're not getting that minute of your life back. I stopped reading at "num=num++…"
This is classic undefined behavior. And I'm tired of telling my students to look up what that means, or to get their information from the ISO standards and *stackoverflow* only. And I'm appalled how people who should know better don't. My point is, programmers need to know what compilers are, what they do and why they do.

See, for us programmers, our ISO standards our scriptures. Our only source of ultimate truth. There are no other axioms. But there are uncertainties even in the holy writs, certain “sins” or “unclear ways of life” about which you ought to know more:

Undefined behavior:
Trying to do certain “wrong” things will result in behavior for which the C standards do not define what must happen. Usually this is because you’re trying to do something stupid, like dereferencing a NULL pointer or accessing an array beyond its bounds. Writing code that result in undefined behavior would mean different compilers are free to do their own thing for your erroneous code resulting in inconsistent behavior for the same code across compilers.

The reason for the standards not defining the behavior for such cases may be one of many, such as

1. The guys over at ISO believe you would be more careful with your code, and they do not see a point in defining the behavior for things like null pointer dereferencing (doing which is pointless and ill-advised for good reason).

2. There are too many cases such as above where bad coding practices lead to code that is semantically confusing / incorrect. Trying to get the vendors to implement defined behavior for all such cases would be a waste of time and counter-productive (wasting precious instruction cycles or compile time. Why do you have num=num++ in your code anyway, friend?)

Unspecified behavior:
This is when your code is not trying to do something wrong, but the code may behave in two or more (correct) ways depending on the compiler. The vendors are not required to document the exact implementation in such cases. Such as, when you call two functions in a single expression, which function gets called first? There is no right way, either way is right and compilers are free to choose as they wish.

```
int result= foo(1) + foo(2);
```

Implementation-defined behavior:

This is also a type of unspecified behavior, but with more restrictions on how the compiler behaves. Here, all vendors are free to implement the behavior, but they must adhere and remain faithful throughout. That is, each vendor may implement the behavior differently from each other, but the compiler itself always results in only one of the many possible implementations. Also, more importantly, the vendor has to properly document the exact behavior in these cases.

An example case would be right-shifting a signed integer.

Let me sign off, there is no moral of the story here. I will meet you in the next edition of “No stupid questions please”.