

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2007-12/msg00442.html>

- *From:* spinoza1111 <spinoza1111@xxxxxxxxx>
 - *Date:* Mon, 31 Dec 2007 11:52:29 -0800 (PST)
-

On Dec 31 2007, 10:00 pm, Richard Heathfield <r...@xxxxxxxxxxxxxxxxx> wrote:

spinoza1111 said:

I have an idea for reducing the cyberbullying at this ng which I've also used at humanities.lit.authors.shakespeare.

It is "The Spinoza Challenge".

I will take a test on a programming topic without special preparation and closed book, and report my score, along with information as to my overall familiarity with the tested platform, language or topic.

My first Spinoza Challenge: beat my score: I just took the C++ test that is available on Spark Notes at <http://www.sparknotes.com/cs/c++fundamentals/review/quiz.html> and received 76%.

I have no particular comment to make on question 1, although I wouldn't have worded the question like that.

Question 2 is unfortunate, since it has two correct answers; although it's obvious which one is wanted, another of the answers is perfectly defensible. Neither is exclusively correct.

Neither were meant to be: cf. "generally"

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Question 3 is trivial.

Question 4 is badly worded but one can get the gist easily enough. Strictly speaking, the assignment operator – *provided that it is not overloaded to do something else* – evaluates the right hand operand and copies the value into the object specified by the left hand operand.

I think we can assume ceteris paribus as to the possibility of overloaded redefinition of operators

Question 5 is broken. None of the specified types guarantees the necessary precision for dealing with twenty decimal digits. The long double type only guarantees ten. One could reasonably argue that any of the integer types mentioned gives an adequate basis for developing a bignum library that could give the required precision.

Bad flaw, I agree. Still, life's unfair. Do you suggest abandoning the challenge or finding new tests? I can search for better tests if you like. I'd say your analyses exhibit fairly good qualifications were it not for your analysis of question 6, which I do not accept.

Question 6 is broken. None of the supplied answers gives the required output, which is:

a c b d

- (A) will give a\t\nb\td
- (B) will give some output dependent on the (unsupplied) values of the objects given in the expression.
- (C) will give acb\nd
- (D) will give ab\nc\td

In the cases of A, C, and D, the backslashes shown here do not introduce escape sequences (which would in any case break their conformance to the requirement), but literally part of the output, as they are themselves escaped.

You may be right. This question may be trash. Still, to be fair to all, we need to all accept these tests as final in order to be able to compare scores, which are not precise measures of qualifications, but even in their imprecision are better than random character and professional assassination.

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Question 7 is trivial.

Question 8 is broken. All of the suggested answers except A exhibit undefined behaviour, and answer A modifies the value of x. There is no correct answer.

- (A) $x++$;
- (B) $x += x--$;
- (C) $x = x++$;
- (D) $x = --x + 1$;

The correct answer is D, and neither it nor any question exhibits undefined behavior in C++, or in usable C compilers.

A adds one to the value of x and returns the value before the increment to thin air. x is changed.

In B, x is stacked and then decremented in place. The stack copy is then added to x. The decrement happens BEFORE the assignment (simple operator precedence). Therefore x is set to $(x-1)+x$, where x is the starter value. The behavior is NOT undefined, as far as I can tell. Please inform me of why you believe otherwise. We can safely assume x is long or int. x is changed. If someone comes up with a value, int or float or double in which it doesn't change, this still makes this the wrong answer because in D (see below) x is unchanged.

By the same reasoning as for B, x is set in C to $(x+1)+x$.

D decrements x, stacks it, adds one, and then replaces decremented x by $(x-1)+1$. No change except perhaps for pathological cases near the end of floating point precision.

Richard, am I missing something? The value is well-defined as long as you commonsensically assume x to be int. The BEHAVIOR is well-defined, period.

Question 9 is trivial.

Question 10 fails to account for pointer comparison, but is otherwise trivial.

In C and C++, pointers are integers and as such comparable

Questions 11, 12, and 13 are trivial.

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Question 14 is ambiguous – it depends on whether you consider 'between' to be inclusive. I don't, so I might disagree with the marker over which answer should be considered correct.

Question 14 does NOT depend on whether the English word "between" is inclusive. In ordinary speech it is used both ways! But the competing answer uses an OR (hope you noticed) and the best answer is A. The ONLY answer is the BEST answer.

Question 15 is trivial.

Question 16 is trivial if you discount the possibility that the break appears inside a switch.

Questions 17 and 18 are trivial, but 18 is very badly worded.

Questions 19, 20, and 21 are trivial.

Question 22 is incorrect. `sqrt` is declared in `<math.h>` but only a complete bozo would define it there.

If we MUST use terms like "bozo", only a complete bozo would use "define" when he means use, dear Richard. The correct answer, I believe, is `#include`. Do inform me why the question is "incorrect".

Questions 23–27 are trivial.

Question 28 has no good answer. There's no reason why you can't write a linked list class (although there's no point, because there's one in the STL). Nor is there any reason why you couldn't use structs to implement a linked list. Nor is there any reason why you can't construct a linked list that can hold only a single type. Nor is there any point in constructing a linked list that can only have a fixed number of elements.

Question 29 has no correct answer. Arrays don't hold types. They *have* types, but they are aggregate types that can store multiple *objects* of a given type. If we agree that that's kinda what the guy meant only he didn't say it very well, we must discount (A) as being incorrect (because they can hold complicated types such as user-defined types). We must also discount (B) because arrays can't hold function types (although they *can* hold function pointer types). And we must discount (C) for the same reason as (B).

Gee, is a function a type? What on earth would it mean to write a

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

function (define it, that is) and then declare an instance of it?

C and C++ functions are not first class objects.

Questions 30 and 31 are trivial.

Question 32 has no correct answer.

No, creating a struct is similar to creating a class. C sucked because you could not hide functionality inside structs in most compilers (you can do so in C Sharp and Visual Basic). structs didn't know it, but they wanted to be objects.

Questions 33–43 are trivial.

Question 44 has no clearly correct answer. (A) is wrong because it doesn't access the base class function. (B) is wrong because it uses a class name in an object context. (C) is wrong unless the code exists within the class of which the function is a member.

B is the best answer of a bad lot. They forgot to tell you that statics are qualified with the name of the class, and not an instance. I do not speak with C++ honcho authority but as a compiler and language designer who asked himself, while taking the test, how a sensible language would expose a static code-only function, that uses no instance variables. Sure, old VB-4 forced you to instantiate to get an effectively "shared" function, but that's because VB-4 sucked.

Questions 45–47 are trivial.

Questions 48–49 seem to use the word "composition" in a way I'm not familiar with. I think of the term as applying to the moving of a

Nor I. I think I hosed them. Here, you'd have to read the Spark Notes notes to prep, and these notes aren't written by geniuses. They are written by high school teachers and graduate students and are nothing more than study aids.

subexpression evaluation into an object that stores the subexpression to make subsequent evaluation of the whole expression simpler and quicker. It is not obvious that the suggested answers have anything to do with this! This may simply be because I'm not as up on C++ as the author of the quiz.

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum

Given the brokenness of the rest of it, however, I do not find that possibility to be terribly likely.

Question 50 is trivial.

I didn't give answers to the ones I thought had no correct answer. I scored 80% (i.e. 40 out of 50).

--

Richard Heathfield <<http://www.cpax.org.uk>>

Email: -<http://www>. +rjh@

Google users: <<http://www.cpax.org.uk/prg/writings/googly.php>>

"Usenet is a strange place" - dmr 29 July 1999

Excellent score, and thanks for your comments.

I think that IF you have used C++ professionally in the past for any significant length of time, and scored only 4 points more than a person who's only occasionally maintained C++, and who now only uses C rarely, you might want to apologize at this time for the many comments you have made about my professional qualifications. When I posted in 2003, you took a vanishingly small number of data points and blew them out of proportion. I'm not going to do that with the questions you got wrong.

By the same ethics, logic, logical ethics, and ethical logic that you shouldn't go about character and professional assassination based on a few lines of code and many stylistic tics of your own, your failure to score high on this test should not be interpreted as meaning that you're not the Expert you want to be. You are...until you incite mob action and enable cyberbullying, because being a professional is more about collegiality and less about examinations and picoseconds.

I ask you in future to emulate my style in my comment on your comment on question 6. I believe that the prefix and postfix increment and decrement operators have, or damned well better have, an order of execution relative to the precedence of assignment. If "modern" C programmers override this in some way, say by overloading, then they are idiots.

But, you're not an idiot. I suggest that your score and your mostly intelligent remarks make you anything but, and I simply ask in future that you might want to be, not more humble, but more collegial.

.

Re: Brian Kernighan, maybe I'm not worthy, maybe I'm scum