

Re: Is this math test too easy?

Source: <http://sci.tech-archive.net/Archive/sci.math/2004-12/2557.html>

From: Richard Henry (rphenry_at_home.com)

Date: 11/28/04

Date: Sun, 28 Nov 2004 14:35:25 -0800

"The Ghost In The Machine" <ewill@sirius.athghost7038suus.net> wrote in message news:cbkp72-pha.ln1@sirius.athghost7038suus.net...

> *In sci.math, Richard Henry*

> <rphenry@home.com>

> wrote

> on Sat, 27 Nov 2004 23:10:44 -0800

> <[WReqd.100055\\$SW3.78409@fed1read01](mailto:WReqd.100055$SW3.78409@fed1read01)>:

>>

>> "Mxsmanic" <mxsmanic@hotmail.com> wrote in message

>> news:m3siq0hu2gr5765se5bdug8jihmhtl1vqp@4ax.com...

>>> *Richard Henry writes:*

>>>>

>>>> *I will return to work Monday morning, programming computers (among other*

>>>> *things) relieved to know that here are no longer any computer errors.*

>>>>

>>>> *What is a "computer error"?*

>>>>

>>>> *Another evasion?*

>>>>

>>>> *I should note here that computers do not make errors (at least,*

>>>> *unless there's a problem with such things as bearing wear, external*

>>>> *temperature control, voltage problems [+], etc.) -- *we* do.*

>>>> *More specifically, the following types of problems are available*

>>>> *for the computer to select from -- assuming it even makes sense*

>>>> *to give the computer a choice here, for computers are about as*

>>>> *smart as an insect, if that. Computers merely do what they're told...*

>>>>

>>>> *[1] The software developer problem may not have correctly understood*

>>>> *the problem during initial setup. This might be termed a*

>>>> *communications glitch; one of the more laughable cartoons*

>>>> *(which I'm having some troubles finding on the Web) involved*

>>>> *a sequence of rather odd swingsets (one of them involved chopping*

>>>> *down the tree and then propping it up!), and at the very end the*

>>>> *swingset "What the customer really wanted" was an old-fashioned*

>>>> *rope & tire affair. (A useful reminder when one gets bogged*

>>>> *down in software technicalities -- if one can find the image.*

sci.math: Re: Is this math test too easy?

- > *The only thing I get is "Osama's Swing Set". Three guesses what*
- > *it looks like and the first two don't count. :-) Maybe it's*
- > *a sign of the times.)*
- >
- > [2] *The software developer may have written code that cannot handle*
- > *a certain type of input, as it was beyond the problem specification.*
- > *For example, in a C++ routine such as*
- >
- > *void normalize(double & vx, double & vy)*
- > *{*
- > *double d = sqrt(vx*vx+vy*vy);*
- > *vx = vx / d;*
- > *vy = vy / d;*
- > *}*
- >
- > *which normalizes a vector, the input vx = vy = 0 will lead to*
- > *anomalies, as one cannot normalize a 0-length vector, and this*
- > *routine won't function properly.*
- >
- > *Another possibility is a dictionary lookup routine that handles*
- > *English words, that is handed a word in, say, Klingon.*
- >
- > [3] *The hardware upon which the software is running may not be compatible*
- > *with the written software. The simplest way to demonstrate this*
- > *is to run a Sparc binary on an x86 machine, then wonder why*
- > *it's not working. (I've occasionally tried to run Sparc*
- > *binaries on HP-UX machines, though it's been awhile. It's*
- > *rather quickly evident that such won't work.)*
- >
- > *More subtle errors may ensue in the driver area, in*
- > *which a hardware piece is instructed to do something*
- > *and the software expects a response, but the hardware*
- > *does something else entirely as the software was either*
- > *mis-written or the hardware is having a problem beyond*
- > *the software's capabilities. The result could totally*
- > *freeze the system.*
- >
- > [4] *The user of the software may have forgotten to Read The Fine Manual.*
- > :-)
- > *(Assuming the programmers bothered to write one. :-) This*
- > *is Yet Another Communications Glitch.)*
- >
- >
- >> *Let us return to your earlier optimistic statemetn, that "computer*
- >> *programmers long ago eliminated problems with ambiguity, and they did it*
- >> *in*
- >> *ASCII".*
- >
- > *The programmers didn't eliminate problems with ambiguity from the*
- > *human side, although the ambiguity has largely disappeared from*
- > *the computer side -- if there was any to begin with. (There*

sci.math: Re: Is this math test too easy?

- > are some issues with multiprocessor systems nowadays, if the
- > programs running on the two processors want to interact.)
- >
- >>
- >> Did you mean to imply by that statements that all contents of
- >> computers are encoded in ASCII?
- >
- > All contents of computers are not encoded at all. To the computer,
- > it's an unending stream of switching transistors processing
- > charge packets, driven by a master clock and the occasional
- > signal transition indicating an interrupt from another interface
- > card. One can think of it as a bunch of bits, but to a
- > transistor it's simply a voltage on a gate, or perhaps a
- > current pulse through the collector. (Digital inverters are
- > overdriven amplifiers.)
- >
- > There is the VGA text display mapping, admittedly, and I'd have
- > to look up the details on that; in cards of old 8-bit patterns
- > (with an additional 8 bits as a modifier to specify foreground
- > and background colors, and maybe blinking depending on another
- > register setting somewhere) were mapped to an 8x8 raster block,
- > which had specific patterns of 64 bits each. These bits, when
- > arranged in the standard fashion, look an awful lot like the
- > standard Western alphabet:
- >
- >
- > ..*....
- > ..**...
- > .*...*
- > *...*
- > *.....* = 01000001, for example.
- > *****
- > *...*
- > *.....*
- >
- > A variant of this encoding still exists in cards today, although
- > one can vary the raster block size and the bits stored
- > (card-loadable VGA fonts).
- >
- > What meaning does it have, exactly? To the computer, not much;
- > it simply does the mapping (and this isn't even part of the
- > main processor proper; this is a function of the display electronics).
- > One could just as easily have defined a bit pattern for 01000001
- > to be a glyph in Hebrew, Arabic, Klingon, or Dalek (if the Daleks
- > even have a written language as such, but that's a discussion for
- > quite another newsgroup... :-)
- >
- > There are also routines that are fed addresses of
- > bitstreams which interpret said bitstreams as characters
- > to display; the one that comes to mind is XDrawString()
- > [I forget the Windows variants]. In this case, the
- > library routine is doing the mapping, and one hopes

sci.math: Re: Is this math test too easy?

- > *it's done correctly -- and if not, that's Yet Another*
- > *Communications Glitch.*
- >
- > >
- > > *Or did you mean, perhaps, to imply that all computer programs are*
- > > *encoded in ASCII?*
- >
- > *At some level, most computer programs are encoded in ASCII. This*
- > *is mostly for human benefit; the computer simply receives, at*
- > *the end of the compile/link, a sequence of bits -- and even that*
- > *isn't quite finished, as the kernel has to dynamically reconcile*
- > *traps from the user side. However, the compiler (more properly,*
- > *the software engineer(s) writing it) has to understand*
- > *in part the human side of the equation, generating meaningful*
- > *(to the developer) diagnostics.*
- >
- > *source code ==> object code ==> executable*
- > *compiler linker*
- >
- > *The problem is that the executable/object still isn't finished,*
- > *in modern computers; a dynamic loader needs to add in still*
- > *more functionality (usually, shared library routines such as*
- > *open(), fork(), and mmap(), which are present in almost*
- > *every Unix system today; even Windows has variants of these).*
- >
- > *dynamic loader*
- > *(ld.so on Unix/Linux)*
- > *executable !=> in-memory image ==> kernel trap handler*
- > */ trap*
- > *library code ==/*
- > *(.so or .sl files)*
- >
- > *Windows has a similar mechanism using .DLLs, but I forget the details.*
- > *(A note on "in memory", which for virtual memory machines can get*
- > *rather funny; a page of code may actually be sitting on disk,*
- > *if the process doesn't need it. Of course processes are themselves*
- > *abstractions as well; it's all one microprocessor trying to*
- > *do everything...)*
- >
- > *Windows has also introduced MSIL, which I know very little about*
- > *beyond its existence. I'm not quite sure where it fits into*
- > *the above, but at some point, when Microsoft gets a round tuit,*
- > *and everything is written in <insert random language here>#,*
- > *Windows can finally occupy all machines in the world for which*
- > *there exists an MSIL -> machine level translator, as Windows*
- > *is destined to do if one believes the documentation... :-)*
- >
- > *(Personally, I hope Linux or FreeBSD gets there first. Linux*
- > *already supports nearly two dozen machine types, and some of*
- > *them have subtypes -- e.g., I know m68k supports both Amigas*
- > *and Ataris. NT in its heyday might have supported 3:*

Re: Is this math test too easy?

sci.math: Re: Is this math test too easy?

- > X86, Alpha, and PPC. For its part I'd have to look to see
- > how many machine types FreeBSD supports; I don't have it here.)
- >
- > In times of yore, sans shared libraries, the executable was
- > complete; it was loaded into physical memory and, when it
- > needed to read from a disk, it would issue a trap. (The
- > PDP 11/xx instruction for this was probably EMT, but I've
- > frankly forgotten the details now. Modern Linux/x86 systems
- > use INT \$80/INT 80H for much the same purpose, though programs
- > outside of such places as glibc actually using such traps
- > explicitly are (and should be) very rare. Note that Linux on
- > other hardware will use other instructions specific to that
- > hardware -- one of the reasons why explicitly specifying
- > the traps can lead to Yet Another Communications Problem.)
- >
- > The kernel program, which is always resident in memory,
- > receives the trap, checks to see that it's legitimate (user
- > programs aren't always perfectly written, and may request
- > invalid memory addresses or not have privileges for certain
- > operations such as raw disk I/O), performs the requested
- > operation, and returns a status code after modifying the
- > state of the system in an appropriate fashion.
- >
- > (Appropriate, in this case, is defined by the system designer.
- > Obviously, open() for instance opens a file -- which makes its
- > data available to the user program, and may make the data
- > modifiable as well, permissions permitting. Of course,
- > Unix was OO before there *was* an OO: open() could also enable
- > communications through a serial device, and the program wouldn't
- > know the difference unless it tried to do an inappropriate operation
- > (e.g., lseek(), mmap()) on the resulting opaque fid returned.
- > Internally, one can see very vtbl-like constructs -- basically,
- > tables of function pointers -- in the Linux kernel.)
- >
- >>
- >> To expand on my previous example, the hexadecimal value "30" located in
- a
- >> computer memory can have at least the following meanings:
- >
- > Actually, the way you've written it, it's not hexadecimal at all,
- > but merely a sequence of two ASCII characters. At a lower
- > level, it's a sequence of 16 bits: 00110011 00110000; at
- > an even lower level, it's a series of charges, pits, or
- > magnetic domains, depending on whether one's looking at it
- > as it passes through the wire, on a CD or DVD, or on a
- > standard tape, floppy, or hard drive.
- >
- >> 1) the decimal value 48
- >
- > This is more or less correct.
- >

sci.math: Re: Is this math test too easy?

- > > 2) the decimal value 30
- >
- > This is **not** correct if your spec is correct;

Binary-coded decimal (BCD). Some modern microprocessors even have BCD arithmetic instructions.

- >the "30" character
- > string is being interpreted in **decimal** here. This is a
- > communications problem.
- >
- > Yet another interpretation is the numeric value 24, if the
- > system interprets "30" in octal e.g.,
- >
- > int n;
- >
- > sscanf("30", "%o", &n);
- >
- > Or one can interpret the character string as one of the values
- > 13104 or 12339, depending on the endianness of one's machine.
- > 13104 = 0011001100110000(2), 12339 = 0011000000110011(2). Such
- > an interpretation is very useful during RSA-style encryption,
- > which requires very very long integers -- 1,024 bits as opposed
- > to the 16 bits here.
- >
- > In most cases, programs will write "030" for a value that is
- > intended to be interpreted in octal; the C compiler in particular
- > sees that leading 0 and says "octal". If one writes "0x30", the
- > C compiler will interpret it as a hexadecimal integer. Certain
- > routines such as strtol() can use a similar convention.
- >
- > It's all communication.
- >
- > > 3) the decimal value 0 (zero)
- >
- > This is correct in certain contexts. For example, the C program fragment:
- >
- > printf("%c", (char) 0x30)
- >
- > will print the single ASCII character "0". The C program fragment:
- >
- > printf("%c", (char) 30)
- >
- > will probably not print out anything interesting, although on old
- > Tektronix scopes strange things might happen; I'd have to look now.
- >
- > > 4) some fragment of a larger decimal value
- >
- > No way to know without more data.
- >
- > > 5) a memory address

Re: Is this math test too easy?

sci.math: Re: Is this math test too easy?

>
> *Unlikely, but possible in older systems (very old C compilers in
> DOS allowed NULL pointers; the value was in fact interpreted as
> an offset into an Intel paragraph). Nowadays, any value
> less than about 4,096 is considered a bad access. However,
> I'd have to look.*
>
> > 6) a fragment of a memory address
>
> *No way to know.*
>
> > 7) an offset from an address contained in another memory location or
> > processor register
>
> *No way to know.*
>
> > 8) an instruction to the computer, the exact meaning varying from
processor
> > to processor
>
> *This is true, and another communications failure may ensue if
> this isn't the "right" instruction -- but the computer can't tell. :-)*
>
> > 9) a fragment of such an instruction
>
> *No way to know.*
>
> > 10) a temporary volatile location used to track the progress of a
computer
> > process (e.g. loop counter)
>
> *A very old computer such as the HP 21xx series [*] might have used such;
> register A, for instance, was memory location 0.
> Register B was 1. (Those were the only two real registers it had.)
> Teletypes could be sent characters by writing to a memory location
> somewhere in the first few words (it was a 16-bit word-address
> machine). Ah, ferrule core.*
>
> *Nowadays, most systems reserve higher blocks of memory than the
> zeropage, if MMIO is used. I'd have to look to see what the PC
> does in that arena, and the common PC is very weird in some areas
> because of evolution -- and because the original PC tried to do things
> in a slightly illogical fashion; in retrospect for example we would
> have been far better off had the engineers written the PC bios to
> simply return a paragraph:offset to the video RAM (and the size of
> said RAM) for programs to play with, instead of trying to
> emulate a teletype (which programmers simply bypassed, writing
> to the video RAM directly, as it was faster than going through
> the INT calling sequence -- and now *everyone* on the planet
> knows about B000:0000/B800:0000/A000:0000, or should :-).*
>

Re: Is this math test too easy?

sci.math: Re: Is this math test too easy?

> > *11) some other value, memory location, or instruction encrypted for security*

> > *purposes*

>

> *"30" is a bit too short to be properly encrypted, although I suppose anyone using ssh over a standard network might have to deal with encryption of single characters.*

>

> >

> > *As for your statement that memory locations are assigned, not encountered,*

> > *what data type and storage size would you assign for an integer? A floating*

> > *point value? A pointer to an integer? A pointer to an array of pointers to*

> > *functions returning floating point values?*

> >

>

> *This is all a communications problem between hardware and software.*

> *In times of yore characters did not have to be 8-bit (one computer had 9-bit chars, for example), and the HP 21xx series had to pack chars in 2 at a time, since there was no concept of byte-addressable memory back then. (One program that needed such simply shifted an address right 1 bit, and checked the carry to see which part of the word was desired. Since the machine could only physically address 32 kwords -- see below -- that more or less worked.)*

>

> *One other interesting problem in the HP 21xx series: if bit 15 was set in an address word, any instruction using that word to refer to its memory (it could support up to 8 kilowords of memory), would do an indirect fetch, which could itself do another indirect fetch, which ... well, you get the idea; occasionally the machine would get corrupted enough to go into an infinite loop of indirect memory cycles, and had to be reloaded, usually from a binary BASIC tape (it was at my high school, many many years ago).*

>

> *Does the engineer know his machine?*

>

> *Of course, nowadays, the code being written has to work on multiple machines.*

> *Some strange things might happen with code such as*

>

```
> void a(int * ptr) { ... *ptr = 1; }
```

>

```
> void b() { short x = 5; a(&x); }
```

>

> *if one's not *very* careful. Compilers like to help here, as well; nowadays using the sequence*

>

```
> printf("%s is a string, right?\n", 0x30)
```

Re: Is this math test too easy?

sci.math: Re: Is this math test too easy?

>
> will elicit a warning diagnostic; GCC in particular spits out
>
> test2.c:5: warning: format argument is not a pointer (arg 2)
>
> It's helpful, but the sequence
>
> char *fmt = "%s is a string, right?\n";
> int val = 0x30;
>
> printf(fmt, val);
>
> will probably not elicit the warning, and fail during runtime instead.
> Fortunately, that sort of usage is fairly rare.
>
> It also helps to know one's language and routines. In C, for example,
> every string is by convention terminated with a \0 (a fancy way of
> specifying an 8-bit zero string within double-quotes). Therefore
> sequences in C such as
>
> char buf[12];
> char *ptr = "Hello world!"
>
> strcpy(buf, ptr);
>
> will *not* work quite right, as buf is a char short. YACP.
>
> There are also useful constants and pseudo-functions; sizeof(short)
> in C, for example, returns the number of bytes a short variable
> occupies, or one can write sizeof(x). Regrettably, there are no
> pseudo-functions for alignment checking, though nowadays it's not
> that big of an issue, as most machines prefer natural alignment:
> 4-byte longwords will be aligned on a 4-byte boundary, 8-byte doubles
> will be aligned on an 8-byte boundary, etc. The compiler supplies
> padding if the user doesn't.
>
> I won't go into much detail beyond mentioning the existence of
> virtual machines, but one could make a case that C, C++, Pascal,
> and such allow for simplification of the software development
> process, and even old-style assembly code abstracted the problem
> slightly (by allowing the programmer to focus on his code logic,
> rather than trying to shift all of his addresses around as he
> modifies/evolves said code).
>
> Also, as a historical note: I've worked on PDP 11/70s (in my
> college daze); an int there would be considered a short by
> modern standards, but held onto a pointer nicely (the address
> range of a process was at most 32,768 bytes in the Unix 6/7 days).
> Nowadays, 64-bit pointers are fast becoming the norm; one has to
> use a "long long" to store those -- if one bothers; most people
> would simply use a "void *" or "something *", where something

Re: Is this math test too easy?

sci.math: Re: Is this math test too easy?

- > *is the item to which the pointer is expected to point*
- > *(int, char, functioncall, struct AnInterestingDataPacket, etc.) I*
- > *could see 128-bit pointers in our future though there is a practical*
- > *limit and/or tradeoff here.*
- >
- > *Know Thy Tools.*
- >
- > *[+] You'd be surprised how many PCs are unreliable because of*
- > *power supply problems.*
- >
- > *[*] just to give you an idea how old: I graduated high school in 1980.*
- > *We called it by the very original name "Num Num".*

That's more or less what I would have said.

Only I would be shorter.