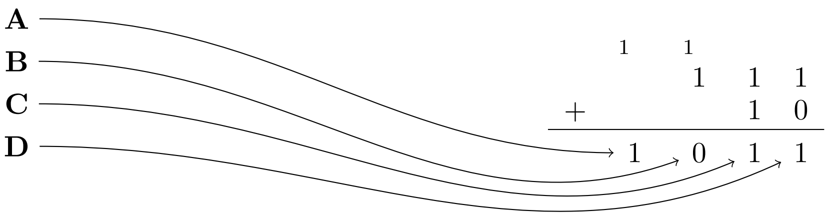
CS 210 Multiple Choice Quiz 1

Alex Vondrak

Spring 2012

1. Which of the indicated digits is incorrect in the following addition of $(\texttt{111})_2$ and $(\texttt{10})_2$?



2. Suppose you're adding two numbers in an arbitrary base, r. If, in some column, the digits (say x and y) sum to a value $\geq r$, what is going to be the value of the carry digit, c?

$$\begin{array}{rl} & \quad c^{\nwarrow} \\ & (\ \ldots \quad x)_r \\ + & (\ \ldots \quad y)_r \\ \hline & (\ \ldots \quad s)_r \end{array}$$

A 1

B r

C $r - 1$

D Depends on what the sum of x and y was to begin with

3. Suppose you're adding two numbers in binary. In any given column, we have digits x (either `1` or `0`) and y (either `1` or `0`). We might also have a value that's been carried "in" from a column to the right, but let's ignore that for now. In the x & y column, we have a sum digit, s, and potentially a c carried "out", but again, disregard it.

$$\begin{array}{rl} & \quad c^{\nwarrow} \\ & (\ \ldots \quad x)_2 \\ + & (\ \ldots \quad y)_2 \\ \hline & (\ \ldots \quad s)_2 \end{array}$$

What is the value of s in terms of a Boolean operator between x and y? Consider `1` true and `0` false. That is, $s = x$ `op` y, where `op` $= \ldots$?

A `AND`

B `OR`

C `IMPLIES`

D `XOR`

4. Which row is the first to contain an error in the following conversion of $(41)_{10}$ to base 3?

	n	$\lfloor n/3 \rfloor$	$n \bmod 3$
A	41	14	2
B	14	4	1
C	4	1	1
D	1	0	1

Result: $(\texttt{1112})_3$

5. Suppose you want to convert a binary number to base 64. How could you do this?

A You can't; we don't have enough symbols

B Use the `base_r` algorithm from page 4 of the notes

C Convert it to hexadecimal, then form groups of 4 hex digits starting from the radix point

D Form groups of 6 binary digits starting from the radix point

6. What is the value of $(\texttt{0.2912})_{10}$ in base 5?

n	$\lfloor n \times 5 \rfloor$	$n \times 5 - \lfloor n \times 5 \rfloor$
0.2912	?	.4560
0.4560	?	.2800
0.2800	?	.4000
0.4000	?	.0000

Result: $\underline{\underline{(\texttt{.????})_5}}$

A $(\texttt{.2121})_5$

B $(\texttt{.1212})_5$

C $(\texttt{.1221})_5$

D $(\texttt{.2112})_5$

7. In a computer, every bit of binary information is stored in a *binary cell*, which can exist in two different states representing either 1 or 0 (high voltage or low voltage). A series of n binary cells together forms a fixed-width *register*. This is the most common device in the computer for storing data, and the basic operations of a computer involve *register transfer*, whereby data shifts from one set of registers to another. E.g., when you hit a key on the keyboard, signals are transfered to an input register, which may be transfered to registers in the processor, which might transfer those values to the RAM (*r*andom *a*ccess *m*emory, the "working memory" of a computer), and so forth.
Suppose you can probe inside a running computer and you find the bits `1100001` stored in some 7-bit register. What value is represented by these bits?

A The ASCII letter `a`

B The unsigned number $(\texttt{97})_{10}$

C The signed 2's complement number $(-\texttt{31})_{10}$

D None of the above

8. How many discrete values could a 7-bit register store?

A 2×7

B 2^7

C Infinitely many

D None of the above

9. Say you're computing $(M - N)$ using radix complement subtraction on the unsigned binary numbers $M = (\texttt{10100})_2$ and $N = (\texttt{1011100})_2$. Which of the following steps is the first to contain an error?

A First, take the 2's complement of N, which is $\texttt{0100011} + \texttt{0000001} = \texttt{0100100}$.

B Add M and the 2's complement of N, giving us

$$\begin{array}{cccccccc} & & & & 1 & & & \\ & & & 1 & 0 & 1 & 0 & 0 \\ + & & 1 & 0 & 0 & 1 & 0 & 0 \\ \hline & & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

C There was no end carry, so take the 2's complement of the sum, which is $\texttt{000111} + \texttt{000001} = \texttt{001000}$.

D Prepend a minus sign to the 2's complement of the sum, giving us the final result: $-\texttt{1000}$.

10. How can you convert a base-10 integer d into signed 2's complement form?

A Use the `base_r` algorithm from page 4 of the notes on d

B Use `base_r` on $|d|$, then prepend a sign bit

C Use `base_r` on $|d|$, prepend a `0`; if $d < 0$, take the 2's complement of the result

D Use `base_r` on $|d|$, take the 2's complement of the result