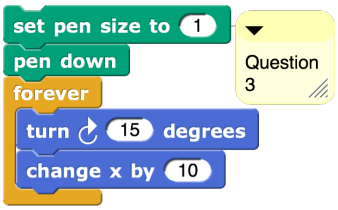
CS10 Fall 2017 Quest Answers

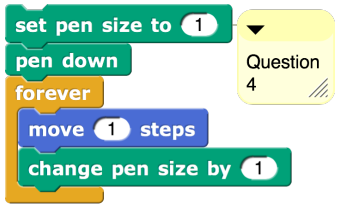
Question 1: Which is *NOT* a benefit of (the computer science definition of) Abstraction? It's easier to debug abstract ideas because they have no concrete form. (This is the common vernacular definition of Abstraction -- opposite of concrete -- but that's different from how it's used in Computer Science, which is to remove detail and generalize.

Question 2: What is the *hex* value of the expression: 10_{10} (decimal) + 101_2 (binary)? 10_{10} is ten, and 101_2 is five, so the sum is fifteen, which is 0xF.

Question 3: If the sprite starts in the middle of the stage facing up & runs the script to the right, what is *eventually drawn on the stage* after the stage stops changing? This was directly from lab, it's a line.



Question 4: If the sprite starts in the middle of the stage facing up & runs the script to the right, what is *eventually drawn on the stage* after the stage stops changing? This draws a triangle, since the pen continues to grow bigger and bigger until it's off the stage.



Question 5: If the output from **Mystery** is true, which can you say *for sure*? A must be true (so the if follows the "then" case), and B must be true (so it reports true).

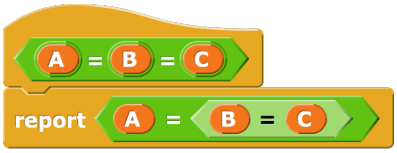
Question 6: You realize you could replace the *entire* body of **Mystery** with a single **report** (as shown below). What could go in there so **Mystery** functions the same for all values of Boolean inputs A and B? A and B (this follows from Question 5)

Question 7: What does it mean for a language to be "Turing Complete"? The language was able to *simulate a Universal Turing Machine*; i.e, it was as powerful as any could be. (straight from lecture)

Question 8: We want to identify the *maximum value* in a list of one or more integers. Consider two versions of the algorithm below. Your job is to identify when they *won't work*. (A) Set a variable **max** to **0**. Iterate through the list of integer values. For each item, if a data value is greater than the value of the variable **max**, set **max** to the data value. (B) Set a variable **max** to the first data value. Iterate through the list's remaining values. For each item, if a data value is greater than the value of the variable **max**, set **max** to the data value. (B) works for all input, (A) only fails when the maximum value is negative, since the initial value (0) will be larger than every value in the list, and the max will always be 0.

Question 9: What is the running time of the algorithm described in the right column (B)? Linear, since there's constant work per step through the list (just a compare).

Question 10: You write a block to determine whether three values are all equal (when they are, it should return **true**, otherwise **false**). You test it and it's fine:



Your friend thinks it *might* have a subtle bug. Choose the inputs that will reveal the bug (if possible), one in which your program returns **true** but should return **false**, and vice versa.
(for each, select ONE per column, or if you believe that bug can't happen, choose the last row)

- Your program returns **true** but *should* return **false:** To make the program incorrectly return true when it shouldn't, we can either set B and C to be the same (but not "true") and A to be true. Then B=C will be true, and A's value of true will be the same as the result of the B=C test. Or, we could set B and C to be different, so that test will be false, but set A to be false.
- Your program returns **false** but *should* return **true:** To make the program incorrectly return false when it shouldn't, we just simply set A, B and C to be the same (but not "true").

Question 11: We love our powerful list processing tools of map and keep! (This is an abstract visual representation so we can just focus on the blocks themselves, not the details of the inputs). For each of the following, which solution works using the fewest of these blocks? Note that there may have to be non-powerful blocks before and/or after calls to these 3 blocks, we only care about these 3 blocks.

Problem a: Given a list of temperatures in Fahrenheit, return a list of the equivalent temperatures in Celsius. map
Problem b: Given a list of the money in the pockets of all the students, return the total amount. None of these, that's combine +.
Problem c: Given a list of words, return a list with them all sorted alphabetically. None of these, that's complicated. It can be done with a clever combine, but that's quite difficult.

Problem d: Given a list of words, return a list of the lengths of all the words that start with X. map keep