UC Berkeley's CS10 Spring 2019 Midterm Solutions: Prof. Dan Garcia

_____ _____ _____

Your Name (first last) *SID* *Lab TA's Name*

_____ _____

← *Name of person on left (or aisle)* *Name of person on right (or aisle)* →

Fill in the correct circles & squares completely…like this: ● *(select ONE)* ■ *(select ALL that apply)*

What's that Smell? Oh, it's Potpourri! (2 pts each for 1-6, lowest score dropped)

Question 1: What was shared with you in the *Testing + HW3* lecture? (select ONE)
○ *Unit testing* is when you test all your code together (i.e., the top-level function), as *one unit*.
Nope, just the opposite. Unit testing means we test each block individually for correctness according to its spec with example inputs and outputs.
○ *Black-box testing* is when you test only the commands that have no inputs or outputs (just like a black box).
Nope, Black-box testing tests if you have no idea what's inside (treating the block as if you CAN'T see inside, like a black box)
○ *Glass-box testing* is when you only need one test to break the code (which is considered fragile, like glass).
Nope, glass-box testing as if you know what's inside your code (treating the block as if you CAN see its insides, like a glass box).
○ *Regression testing* is when you pretend you've regressed to a younger age, and test with nonsense inputs.
Nope, regression testing is a process where you add a feature and then re-run through all the old tests.
● None of these

Question 2: What was shared with you in the *Computing & the Environment* lecture? (select ONE)
○ Even though *80% of E-waste is properly recycled*, the other 20% is many millions of tons of waste!
Nope, it's the opposite. 20% is *of E-waste is properly recycled*, the other 80% is many millions of tons of waste
● *Affected people in countries receiving e-waste have become dependent on it*, even though it's killing them.
Yes, sadly.
○ The *appetite for compact discs*, with more and more people having disposable income (and access to Amazon and similar delivery services that can ship them to you so easily), *hasn't stopped growing*!
Nope, it's the opposite. CD sales peaked and have been declining for years.
○ *Nobody has found any real uses for old cell phones*, since their hardware/software is so outdated.
Nope, it's the opposite. Faculty have created projects that have used cell phones as mini-clusters, as well as acoustical sensors in the forest.
○ None of these

Question 3: What was shared with you in the *Computers in Education* lecture? (select ONE)
○ Judah Shwartz classifies the uses of computers in education into: { Individual, Collaborative, Computer-led }.
Nope, it's { Tools, Microworlds, Courseware }.
○ xMOOCs are known for their highly collaborative structure, where folks are as much teachers as students.
Nope, that cMOOCs (c stands for connectivist). xMOOCs are mostly lecture-centric.
○ Sir Ken Robinson believes that education should be based on industrialism for efficiency and better learning.
Nope, it's the opposite! He believes education is *already* based on industrialism and should change and "go in the opposite direction".
● Prof. Brian Harvey argued that standardized testing has changed what counts as knowledge in schools.
Yep, and that was the answer to the clicker question!
○ None of these

Question 4: What was shared with you in the *Concurrency* lecture? (select ONE)
○ Moore's law is the exponential growth in number of cores in CPUs (essentially doubling every 18 months!).
Nope, Moore's law describes the exponential growth of transistors on an integrated circuit (IC).
○ In 2005, a sea change in computers happened, and we couldn't keep making parallel computers faster!
We couldn't keep making *serial* computers faster.
● The "sea change" to multi-core meant the computing community had to rethink its languages and algorithms.
Yep! It was even a clicker question…
○ Deadlock can happen with three or more "workers", but not with only one or two "workers".
Nope, deadlock can also happen with only two workers.
○ None of these

Question 5: If 5% of a program is serial, what's the max speedup we can get with ∞ cores? (select ONE)

○	○	●	○	○
5x	10x	20x	95x	None of these

Amdahl's law says the maximum speedup with ∞ cores is 1/s, where s is the fraction of a program that's serial. So if s = 1/20, then speedup = 1/s = 1/(1/20) = 20x.

Question 6: What is $1010_2 \times 2_{10}$? (select ONE)

○	○	○	○	○	○	○	○	○	●	○
2020_{20}	2020_{16}	10100_{16}	$A0_{16}$	24_{16}	22_{16}	20_{16}	18_{16}	16_{16}	14_{16}	12_{16}

$1010_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 1*8 + 0*4 + 1*2 + 0*0 = 8 + 2 = 10_{10}$, which multiplied by $2 = 20_{10}$.
What's 20 in base 16? Make the columns:

16^3	16^2	16^1	16^0
4096	256	16	1

And run the algorithm. How many 4096s fit in 20? 0. How many 256s fit in 20? 0. How many 16s fit in 20? 1 with 4 remaining. How many 1s fit in 4? 4.

16^3	16^2	16^1	16^0
4096	256	16	1
0	0	1	4

Therefore $20_{10} = 14_{16}$

Question 7: *Two keeps are better than one! (…or are they?)* (12 pts)

You are given seven expressions:
This question, like the one on the Quest, is all about Domain and Range. Imagine P is "is the number odd", and Data is a list of numbers. The goal is just to keep only the odd numbers as a list.

a)	 The domain of P is a single number, but DATA is a list. This is a Domain error..
b)	 Similarly, this is also a domain error, but on top of that, P is asked to be able to handle lists (the innermost P) and booleans (the outmost P which is being fed the True/False value from the innermost P). That's a lot to ask of a predicate!
c)	 This calls keep nicely, returning a list of odd numbers, but then calls odd? on that list. Domain error! P only knows about numbers, not lists.
d)	 This calls keep nicely, returning a list of odd numbers. No error!
e)	 For the same reason as (a), this is a domain error since P doesn't know what to do with lists, and even worse, it returns a Boolean which keep errors on, since the "from" slot needs to be a list.
f)	 This calls keep nicely, returning a list of odd numbers. Then that list is fed into another keep, which already told you it was going to keep those numbers, so the outer keep just lets the list pass through passing everyone, and the return value is a list of odd numbers. It's like passing through airport security ONLY to have to pass through it again! You already checked my bag once, I'm sure it'll pass again!
g)	 This takes each number, first passes through the inner P (odd?) with no problems, but then that Boolean is passed into another P. But wait, P's domain is only numbers! (in this case) So that's another domain error.

	a	b	c	d	e	f
b	☐					
c	☐	☐				
d	☐	☐	☐			
e	☐	☐	☐	☐		
f	☐	☐	☐	■	☐	
g	☐	☐	☐	☐	☐	☐

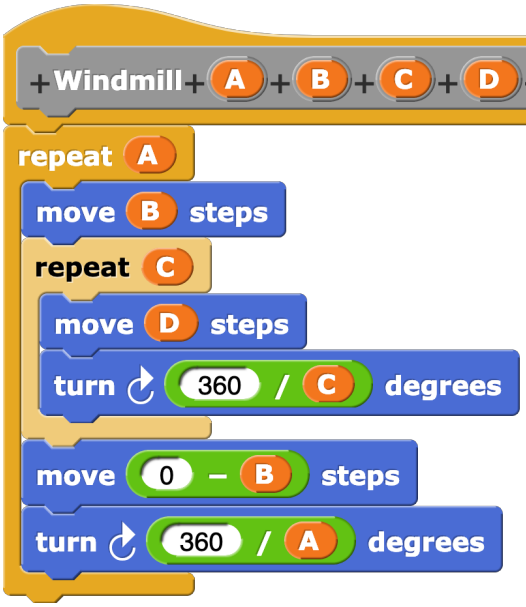
Which pairs are *always equivalent* for **all**  lists and **all** predicates  ? Said another way, which pairs will *always* report the same value? (select ALL that apply; selecting a particular box means you are

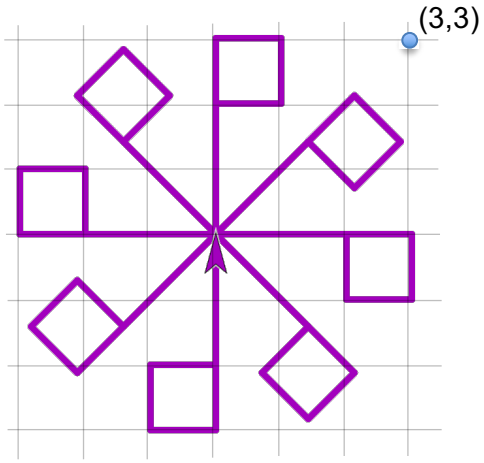
declaring that the expression in the row will always have the same value as the expression in the column **for all input.** *(there is at least ONE, so if you mark no boxes, we'll assume you skipped it and you'll receive no points)*

Question 8: Match each programming paradigm with the description. (select ONE per row, 2 pts)

	Declarative	Object-Oriented	Functional	Imperative
You're not allowed to have any side-effects! Works great with parallelism.	○	○	●	○
Classes are "factories" producing instances; inheritance saves code.	○	●	○	○
Programs are like a recipe: first do this, then that, and next that, etc.	○	○	○	●
Tell the computer *what* you want, not *how* to do it. It works like "magic".	●	○	○	○

Question 9: *Chasing windmills…* (6 pts, 1+1+1+1+2) SID: _____





a) The sprite starts at the origin at (0,0). The upper right grid corner is the point (3,3). The pen is down. As you might imagine, the pen thickness is very small and we zoomed in on the screen. What are the values of **A**, **B**, **C** and **D** for the `Windmill` block that yield the image on the right? (select ONE for each)

A:	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	● 8	○ 9	○ 10	○ None of these
B:	○ 1	● 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	○ 10	○ None of these
C:	○ 1	○ 2	○ 3	● 4	○ 5	○ 6	○ 7	○ 8	○ 9	○ 10	○ None of these
D:	● 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	○ 10	○ None of these

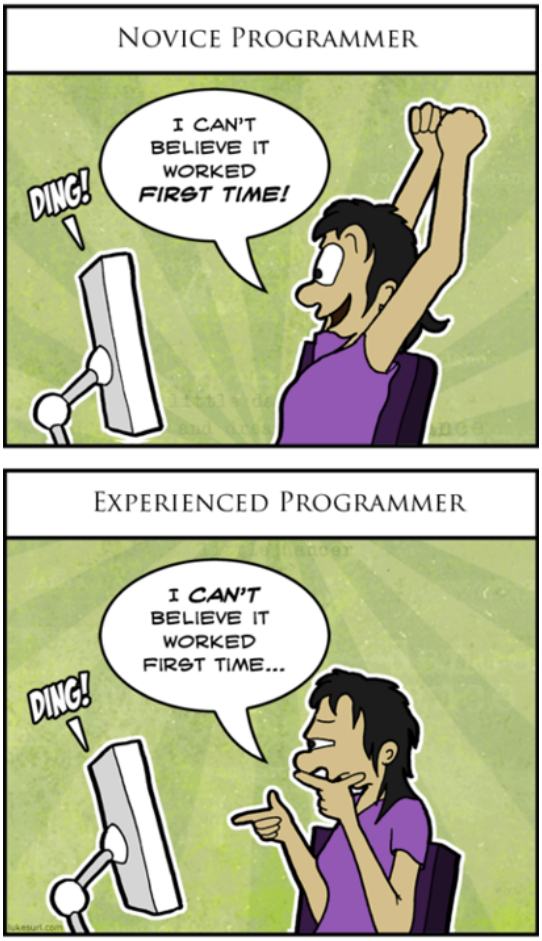
The outer repeat (**A**) controls how many "blades" of the Windmill, and there are 8 of them. You go out 2 steps and draw the fan part of the Windmill, so **B** is 2. Each blade is controlled by the inner repeat, so since it's a square, it's **C** is 4. The side length of each square is 1 so **D** is 1.

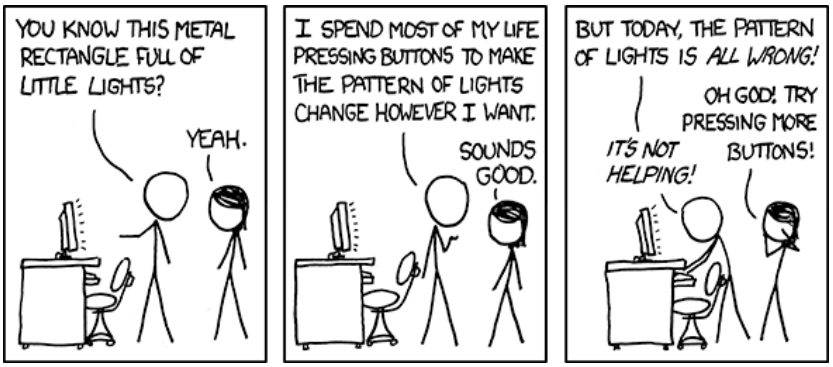
b) What is the running time of 

(as a function of **N**)?
Assume moving and turning take constant time. (select ONE)

This would mean you have N blades, and each blade has N sides (i.e., is an N sided polygon).
So that's N*N=N^2 lines to make all the blades, plus a little more work to move in and out and turn to the next blade but at a high level it's N^2 lines or a Quadratic running time.

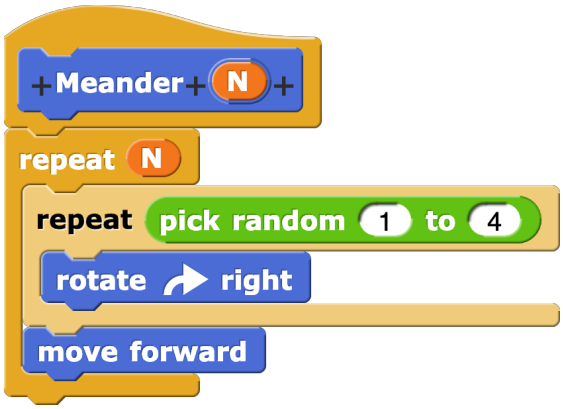
○	○	○	●	○	○
Constant	Logarithmic	Linear	Quadratic	Cubic	Exponential

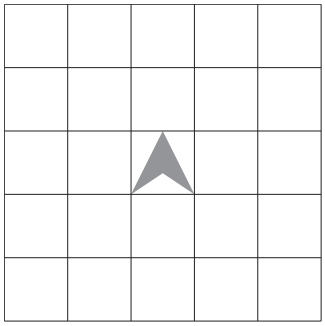


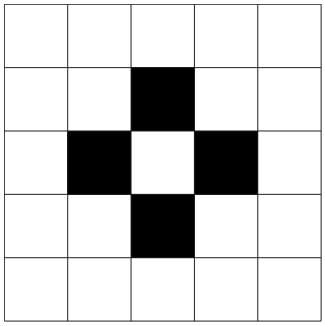


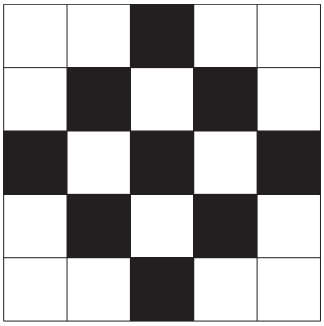
Question 10: どうもありがとうミスターロボット ***Dōmo arigatō, Mr. Roboto…*** (10 pts, 3+5+2)
Here are helper blocks for controlling a robot (shaped like an arrowhead) on a grid world, looking down on it.

move forward	rotate right
The robot moves *one square* forward in the direction it's facing.	The robot turns right, in-place. (That is, it makes a 90º right turn)









For both (a) and (b), **we start with the robot in the middle of the grid, facing up,** as shown above.

a) In the grid above, shade in all the squares that the robot could **end up in** after a call to Meander 1.

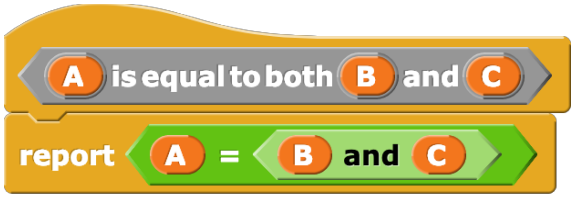
b) In the grid above, shade in all the squares that the robot could **end up in** after a call to Meander 2.

c) What is the running time of Meander N? (select ONE)

○ Constant	○ Logarithmic	● Linear	○ Quadratic	○ Cubic	○ Exponential

There's only one repeat loop (as a function of N) to do something four times, so that's 4N rotate right, which if you plotted it would be a function.

Question 11: ***Inside, we're all the same!*** (8 pts, 2 each)



You author the following (possibly buggy) code because you want to check if **A** is the same as **B**, *and* **A** is also the same as **C.** That is, whether **A** is equal to both **B** and **C**.

For the following cases, choose the appropriate values for **A**, **B** and **C**. *(There may be multiple right answers)* (For each (a)-(d), select ONE per row, or select "This is impossible to achieve!" if it can't be done)

a) A is equal to both B and C is supposed to return **false**, and *does* return **false**.

	true	false	This is impossible to achieve!
A	○	●	
B	●	○	○
C	●	○	

Also, **A** is True, **B** is False, **C** is False, **A** is True, **B** is False, **C** is True, and **A** is True, **B** is True, **C** is False,

b) A is equal to both B and C is supposed to return **true**, and *does* return **true**.

	true	false	This is impossible to achieve!
A	●	○	
B	●	○	○
C	●	○	

Also, **A** is False, **B** is False, **C** is False

c) A is equal to both B and C is supposed to return **false**, but returns **true**.

	true	false	This is impossible to achieve!
A	○	●	
B	●	○	○
C	○	●	

Also, **A** is False, **B** is False, **C** is True

d) A is equal to both B and C is supposed to return **true**, but returns **false**.

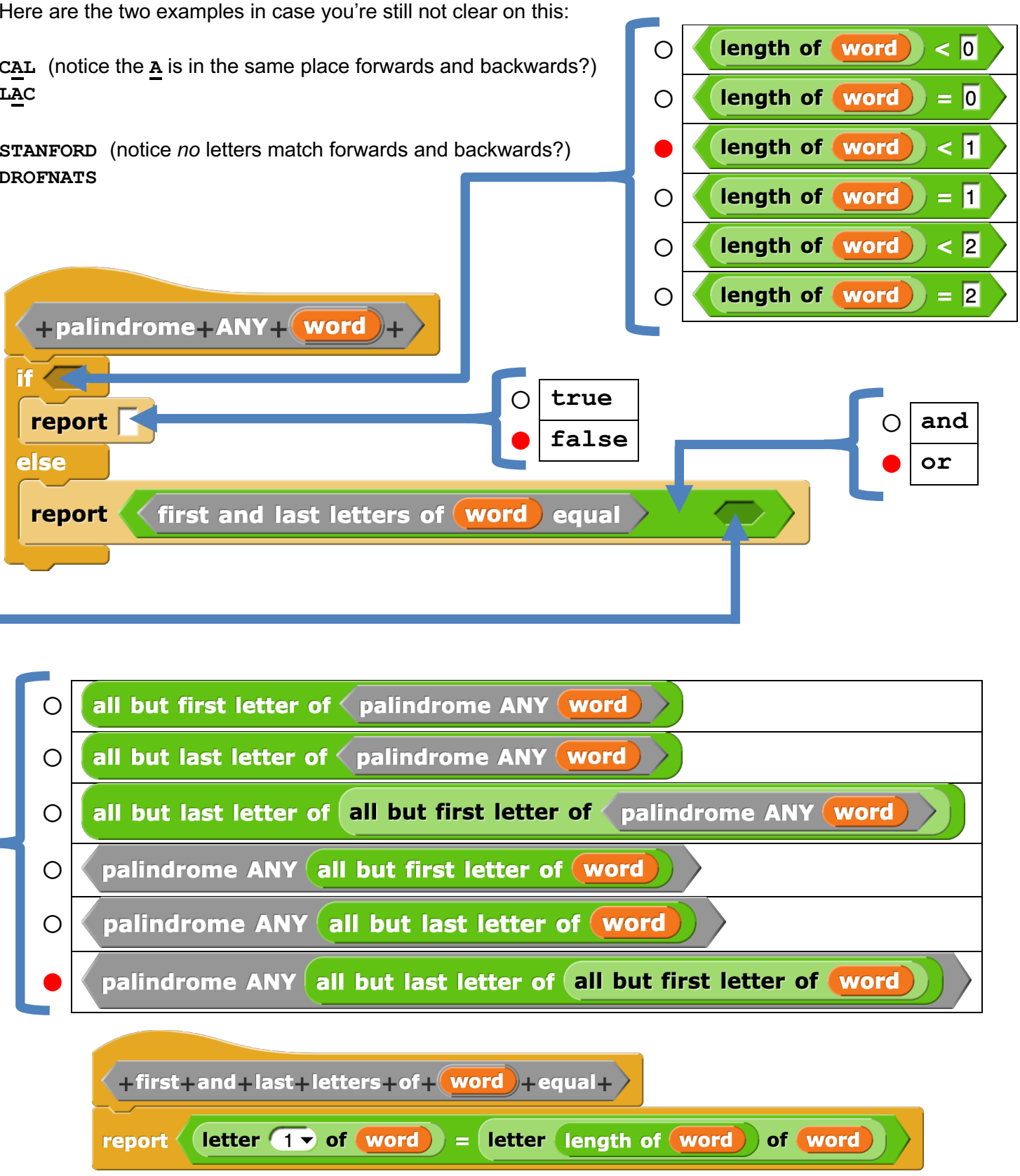
	true	false	This is impossible to achieve!
A	○	○	
B	○	○	●
C	○	○	

Question 12: Why isn't the word *palindrome a palindrome? Awww!* (12 pts)

Palindromes are words which read the same backward as forward, e.g., **OTTO**, **I** and **ANA**. A better name for these words would be *PalindromeALL*, because ALL letters have to match backward and forward. We're interested in *PalindromeANY* words, in which ANY letters can match, e.g., **OTTO**, **I**, **ANA**, **CAL** (note the **A** in the middle matches both forwards and backwards) and **REAR**. You guessed it, **STANFORD** is not a *PalindromeANY* word, since no letters match forward and backwards. Fill in the 4 sets of "select ONE" options to write it. Here are the two examples in case you're still not clear on this:

CAL (notice the **A** is in the same place forwards and backwards?)
LAC

STANFORD (notice *no* letters match forwards and backwards?)
DROFNATS



The way to think about this – the recursive case has to be the last one since it's going to DIVIDE (make the problem smaller by cutting off the ends) and then INVOKE itself on the smaller one.

Since it's looking for ANY matching pair of front-back letters, it has to be **`or`** since it's effectively saying "do the first and last match" OR do the second and second-from-last match OR …

Now, we recurse by removing two letters at a time. So, none of the **`length =`** blocks can be correct, since we'll have odd- and even-number of character lists, and one of those would miss the base case. (E.g., if it were length-of(word) = 1, then if you put in an even-number-of-character word – say length 4 like LOVE, then after one recursion it'll be OV and then it'll be empty, and then cause problems when we continue to ask for all but the first/last letter of an empty word).

Now, as we're recursing down, if we gave it LOVE, we would recurse down with OV, then nothing (length = 0) and in that case it should return **`false`**, since it didn't find it. So it needs to return **`false`**, but what's the stopping case, length < 2 or length < 1 or length < 0? If the length = 1, then it should return **`true`** because a single letter matches that letter when it returns. So that case should return **`true`**. So therefore it can't be length < 2 (because that would mean the length = 1 catches our base case which we just said needs to return **`false`**. It can't be < 0 because then zero-length words would go to the recursive case and you can't ask for the first and last character of a zero-length word. So it's length < 1 which means that empty words will be **`false`**, and 1-length words will have their first letter = their last letter, and then or has found its **`true`** (or is a **`true`** -finder, recall) and it return **`true`** without recursing (and causing an error as it tries to take the first and last letter off of a single letter)!