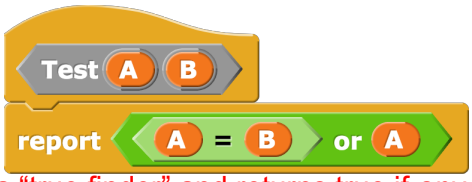
CS10 Fall 2018 Midterm 2 Answers

(The block on the right is used for Questions 10 & 11; 2 pts each)

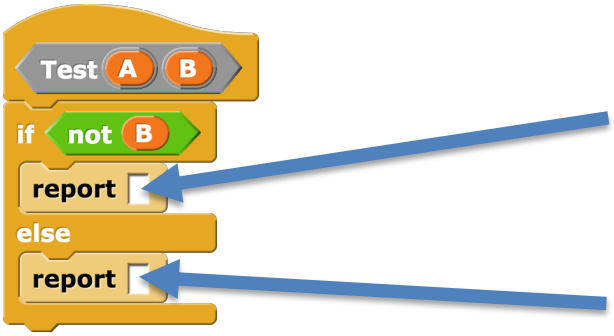


Question 10: If the output from **`Test`** is false, which can you say *for sure*? **`A`** and **`B`** are Booleans. (select ALL that apply)

If **`Test`** is false, then both terms of the **`or`** must be false, since **`or`** is a "true finder" and returns true if *any* of its inputs are true. So we know A must be false. We also know that A=B is false, so that means A ≠ B, and if A must be false, then B must be true!

☐ **`A`** must be true	■ **`B`** must be true	■ **`A`** must be false	☐ **`B`** must be false	☐ None of these

Question 11: Fill in the blanks so the predicate is the same as the original **`Test`** block. (select ONE from each)



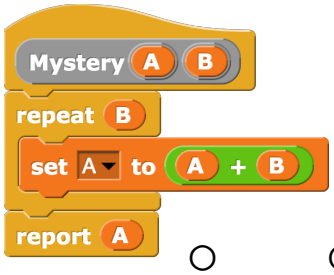
● **`true`**	○ **`false`**	○ **`A`**	○ **`not A`**	○ **`B`**	○ **`not B`**

○ **`true`**	○ **`false`**	● **`A`**	○ **`not A`**	○ **`B`**	○ **`not B`**

So if **`Test`** only returns false when A is false and B is true, then it returns true all other cases. The initial "if not B" then case is only reached if B is false, which is when **`Test`** is supposed to return true (since it's not the A false B true case), so the first report should be true. In the else case, that's when B is true, but we know that if A is false it returns false (we learned from the last problem), and if A is true it's true, so we just return A.

Question 12: What does **`Mystery`** report, if B is a non-negative integer (i.e., 0, 1, 2, …)? (select ONE, 4 pts)

A is incremented with the value B for B iterations, so the first time it's A+B, then it's A+2B, then A+3B, … until it's A+B*B = $A+B^2$



○ $A+B$	○ $A \times B$	○ A^B	○ B^A	○ B^2	○ $A+B^B$	○ $A+B^A$	● $A+B^2$	○ The sum of all the numbers from A to B	○ Error	○ Infinite Loop

Question 13: What is $256_{10} + 10000_2$? (select ONE, 2 pts) *Hint:* $16_{10} \times 16_{10} = 256_{10}$

We're asked to convert these numbers to hexadecimal. Hexadecimal of 256_{10} is 100_{16}, since the columns of hex are $16^3 = 4096_{10}$ | $16^2 = 256_{10}$ | $16^1 = 16_{10}$ | $16^0 = 1_{10}$. The number 10000_2 is 10_{16}, and 100 + 10 (in any base) is 110.

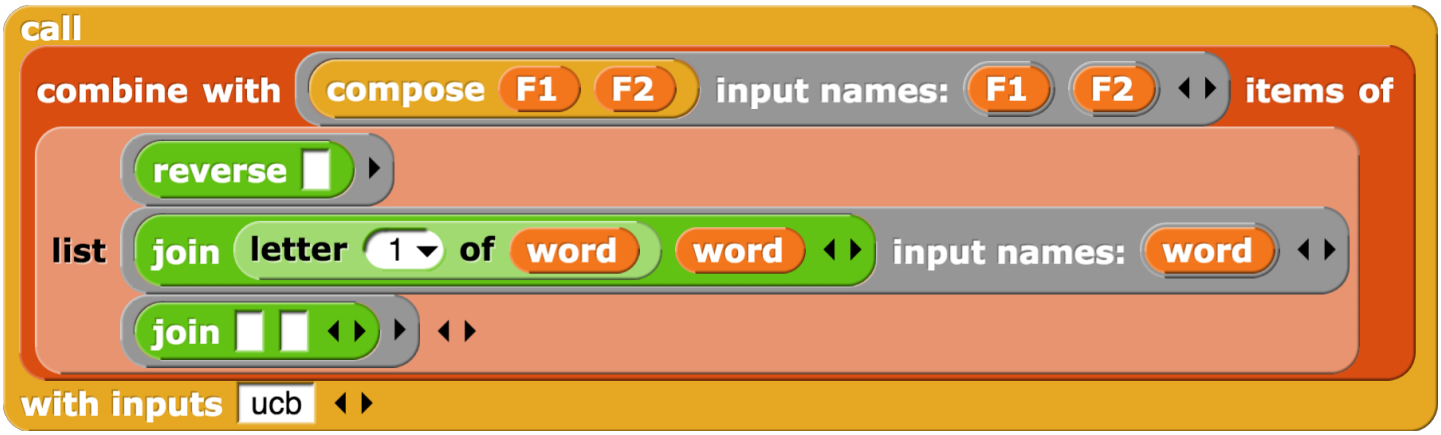
○ AF_{16}	○ FA_{16}	○ FF_{16}	● 110_{16}	○ 111_{16}	○ 210_{16}	○ 10256_{16}	○ 12560_{16}	○ 22560_{16}	○ None of these

Question 14: *Potpourri Part II* (10 pts=2*5) SID _____

c) What does the **combine** expression return? (Choose ONE)

This came straight from lecture, combine (joinswap) items of (a b c d) = a joinswap b joinswap c joinswap d. Let's evaluate it using a right-associative model. That's (a joinswap (b joinswap (c joinswap d))). That becomes (a joinswap (b joinswap dc)) which becomes (a joinswap dcb) which becomes dcba. Now let's evaluate it in a tournament-style manner: (a joinswap b) joinswap (c joinswap d), which is ba joinswap dc, which is dcba. joinswap is an associative function!

○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●
abcd	abdc	acbd	acdb	adbc	adcb	bacd	badc	bcad	bcda	bdac	bdca	cabd	cadb	cbad	cbda	cdab	cdba	dabc	dacb	dbac	dbca	dcab	dcba



d) What does the expression above return, taken straight from lecture with a different input? (Choose ONE)

The combiner creates a frankenstein function, in which the list (f(x) g(x) h(x)) becomes f(g(h(x))). So this is reverse(stutter(duplicate(ucb))) ➔ reverse(stutter(ucbucb)) ➔ reverse(uucbucb) ➔ bcubcuu.

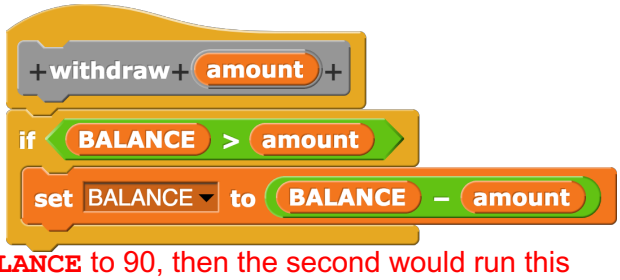
○ laclacc	○ bcu uucb ucb ucb	○ bcuubcuu	● bcubcuu	○ bbcubbcu	○ bbcubcu

c) The developer of Snap*! removes the restriction* that two scripts cannot run at the same time, claiming it will increase performance. *What could now happen?* Note: this problem is independent of the block below. (Choose ALL that apply)

This opens the floodgates to all the concurrency problems that come up with parallel and distributed computing, unfortunately, yikes!

- ☐ Abstraction
- ■ Deadlock
- ■ Livelock
- ■ Race Condition
- ☐ Turing Completeness

e) In fact to show this, you set up a fake bank with $100 in it, and have TWO people simultaneously take $10 out of their accounts using the block above. *What are the possible values of* **BALANCE** *afterward?* (choose ALL that apply)



If they run this one after the other, the first one would withdraw money setting the **BALANCE** to 90, then the second would run this and set the **BALANCE** to 80. If they happened to run at the same time, they could both read the value of **BALANCE** to be 100 *at the same time*, and both then set the **BALANCE** to 90. Hey, a free $10!

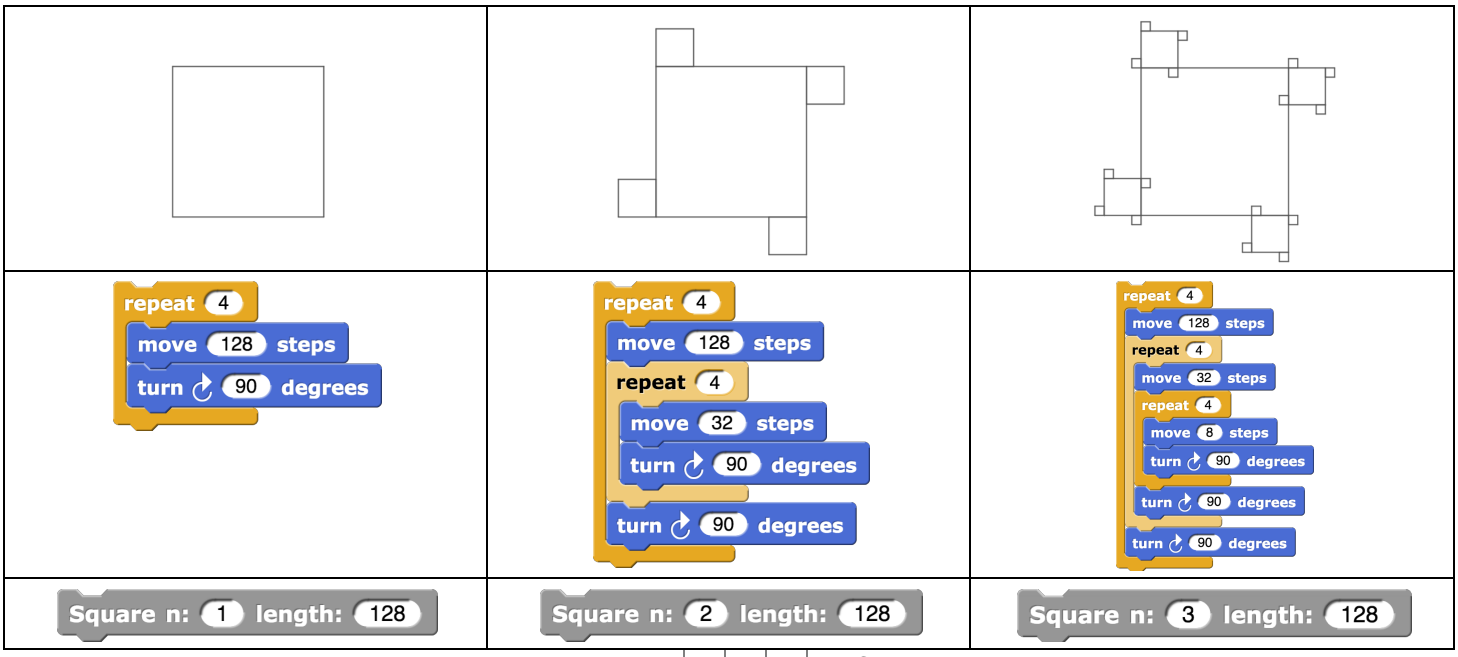
☐ $0	■ $80	■ $90	☐ $100	☐ $110	☐ $120

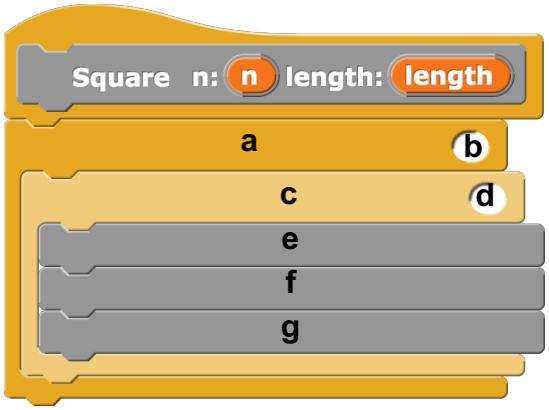
g) In computational science, *computers are used to understand things that are* _____ *for experiments*: (choose ONE)

○ too data-intensive	○ too trivial	○ too cheap	● too slow	○ too experimental	○ too random

Question 15: *We put the fun in functional programming…* (10 pts)

We start with our standard square and add a fun flourish before we make our turn. The sprite starts at the top left of the biggest square facing right. Code and pictures.





Fill in the slot in the row and column corresponding to the expression or block you'd like to place in the code below. Slots **b** and **d** are round but they can take a hexagonal-shaped predicate if that's what you need. (Select ONE per column; you might not need all rows).

a	b	c	d	e	f	g	
○	○	●	○	○	○	○	repeat
●	○	○	○	○	○	○	if
○	○	○	●	○	○	○	4
○	○	○	○	○	○	○	n = 0
○	○	○	○	○	○	○	n = 1
○	●	○	○	○	○	○	n > 0
○	○	○	○	○	○	○	n > 1
○	○	○	○	●	○	○	move length steps
○	○	○	○	○	○	○	move n steps
○	○	○	○	○	○	●	turn 90 degrees
○	○	○	○	○	○	○	Square n: n – 1 length: length
○	○	○	○	○	○	○	Square n: n length: length
○	○	○	○	○	○	○	Square n: n + 1 length: length
○	○	○	○	○	●	○	Square n: n – 1 length: length / 4
○	○	○	○	○	○	○	Square n: n length: length / 4
○	○	○	○	○	○	○	Square n: n + 1 length: length / 4