UC Berkeley's CS10 Fall 2019 Midterm: 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)

smbc-comics.com

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)
- ○ You should try to use *mutable data* whenever possible, since it makes testing easier.
- ○ You should try to use *immutable data* whenever possible, since it makes testing easier.
- ○ Putting all your code in one monolithic, top-level script *is a good thing*, since it makes testing easier.
- ○ You can prove that a block *with state and finite inputs* is correct by testing it on all the possible inputs.
- ○ None of these

Question 2: What was shared with you in the *Computing & the Environment* lecture? (select ONE)
- ○ E-waste is worth $62.5 billion/yr, so *first-world nations are competing to receive and process it*.
- ○ Thanks to streaming services (e.g., Netflix and Spotify) *overall global e-waste emissions is decreasing*.
- ○ Researchers are using old cell phones for *bio-acoustic monitoring of the ocean*, to hear whales.
- ○ Researchers are using old cell phones as a *low-power, low-cost distributed computing cluster*.
- ○ None of these

Question 3: What was shared with you in the *Computers in Education* lecture? (select ONE)
- ○ Using Judah Schwartz' definitions, *Snap! would be a Microworld*.
- ○ cMOOCs are "classroom-style" MOOCs, where lectures from the world's best lecturers are emphasized.
- ○ Prof Harvey: "The most important use of computers in education is web search to access information."
- ○ Sir Ken Robinson believes that we should have more standardized testing, since that brings more efficiency.
- ○ None of these

Question 4: What was shared with you in the *Concurrency* lecture? (select ONE)
- ○ *Amdahl's law* predicted the number of transistors on a chip would double every two years.
- ○ *Moore's law* said that the maximum speedup is a function of the percent of serial code you have.
- ○ *Time sharing* is a technique to allow multiple CPUs to share the work for a same task (a single thread).
- ○ If four self-driving cars arrive at a four-way stop simultaneously, and nobody moves, that's a *race condition*.
- ○ None of these

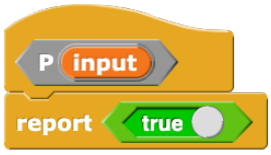
Question 5: If the max speedup with ∞ cores is 5x, what percentage of the code is serial? (select ONE)

○	○	○	○	○	○	○	○	○	○
5%	10%	20%	25%	30%	40%	50%	75%	95%	None of these

Question 6: What is $12_{16} \div 11_2$? (select ONE)

○	○	○	○	○	○	○	○	○	○	○
1_{10}	2_{10}	3_{10}	4_{10}	5_{10}	6_{10}	7_{10}	8_{10}	9_{10}	10_{10}	None of these

Question 7: *Two keeps are better than one! (…or are they? Bwahaha...)* (10 pts)

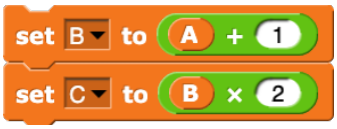
You are given the following predicate 

and list DATA 

.
What do the following expressions return?

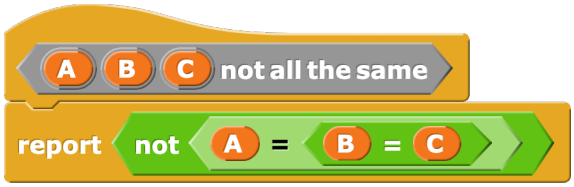
	true	1 Go 2 Bears length: 2	1 true 2 true length: 2	Error
map P over DATA	○	○	○	○
keep items P from DATA	○	○	○	○
keep items P P from DATA	○	○	○	○
P keep items P from P DATA	○	○	○	○
keep items P from keep items P from DATA	○	○	○	○

Question 8: Match each programming paradigm with the description. There should be only one per row and one per column, so if some rows match more than one column, adjust it so they all work. (4 pts=1+1+1+1)

	Declarative	Object-Oriented	Functional	Imperative
	○	○	○	○
	○	○	○	○
	○	○	○	○
	○	○	○	○

Question 9: *Inside, we're not all the same!* (8 pts, 2 each) SID: _____

You author the following (possibly buggy) code because you want to return **true** when **A**, **B**, *and* **C** are not all the same. That is, return **false** only when **A**, **B**, *and* **C** are all **true** or all **false**.



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 "Impossible to achieve!" if it can't be done)

a) A B C not all the same is supposed to return **false**, and *does* return **false**.

	true	false	Impossible to achieve!
A	○	○	
B	○	○	○
C	○	○	

b) A B C not all the same is supposed to return **true**, and *does* return **true**.

	true	false	Impossible to achieve!
A	○	○	
B	○	○	○
C	○	○	

c) A B C not all the same is supposed to return **false**, but returns **true**.

	true	false	Impossible to achieve!
A	○	○	
B	○	○	○
C	○	○	

d) A B C not all the same is supposed to return **true**, but returns **false**.

	true	false	Impossible to achieve!
A	○	○	
B	○	○	○
C	○	○	

Question 10: ***Folks, please line up by SID, smallest to largest…*** (12 pts, 3*2+6*1 pt each)

Students are asked to stand and line up by student IDs (SIDs), **smallest in the front and largest in the back**. Here are 3 algorithms to find out if they are or not. For all problems, assume the number of students (N) is a power of 2 minus 1 (e.g., 2^1-1=**1**, 2^2-1=**3**, 2^3-1=**7**, 2^4-1=**15**, 2^5-1=**31**, …) and *really* big. (How big?) Really big. Also, "clock time" is actual elapsed time if you *used a stopwatch to time the algorithm*, **and SIDs are unique.**

Algorithm I – "Everyone" algorithm

1. All at once, everyone (but the person in the back) writes their SID on a piece of paper, puts it in their left hand and hands it to the person behind them over their right shoulder.
2. Everyone (but the person in the front) takes the paper being handed to them in their right hand.
3. The first person sits down.
4. If anyone notices the given SID is more than theirs, they yell "NOT IN ORDER", otherwise they sit down.
5. If the person in back sees everyone sitting (including themselves), they yell "IN ORDER"

Algorithm II – "Divide and Conquer" algorithm

1. You walk to the middle student of the consecutive standing students, and make sure their number is greater than the SID of the person directly in front of them and less than the SID of the person directly behind them (whether standing or seated), skipping the comparison if there is nobody there.
2. If any of these are out of order (front bigger than back), you yell "NOT IN ORDER" and stop.
3. Otherwise, have that student sit down. If there are no remaining standing students, yell "IN ORDER"!
4. Otherwise, go to the consecutive group of standing students in front of the one seated, and ask a friend – you have an infinite amount of non-student friends – to replicate what you're doing for a similar consecutive group of standing students behind the one just seated, and both of you go to step 1.

Algorithm III – "Random" algorithm

1. If no students are standing, you yell "IN ORDER" and stop.
2. You choose a random student from those who are standing.
3. You make sure that SID is bigger than the SID of the person directly in front (whether standing or seated) and less than the SID of the person directly behind them (whether standing or seated), skipping the comparison if there is nobody there.
4. If any of these are out of order (front bigger than back), you yell "NOT IN ORDER" and stop.
5. Otherwise, you ask that student to sit down and go to step 1.

a) Is each *correct* (i.e., always return the correct value, not error or run forever)? (select ONE per row)

Algorithm	Yes	No
Everyone	○	○
Divide and Conquer	○	○
Random	○	○

b) In the WORST case, what's the *number of comparisons* (NOT running time)? If it's actually between two categories, pick the bigger category. E.g., N^4 is bigger than *cubic*, so pick *exponential*. (select ONE per row)

Algorithm	Constant	Logarithmic	Linear	Quadratic	Cubic	Exponential
Everyone	○	○	○	○	○	○
Divide and Conquer	○	○	○	○	○	○
Random	○	○	○	○	○	○

c) H*ow much clock time* (NOT running time) would each take in the WORST case? (select ONE per row)

Algorithm	Constant	Logarithmic	Linear	Quadratic	Cubic	Exponential
Everyone	○	○	○	○	○	○
Divide and Conquer	○	○	○	○	○	○
Random	○	○	○	○	○	○

Question 11: どうもありがとうミスターロボット ***Dōmo arigatō, Mr. Roboto…*** (16 pts, 2+2+2+10)

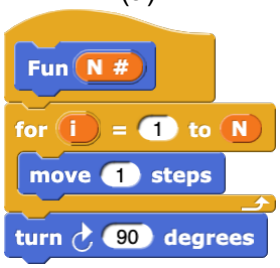
(Clarification: if the sprite were at (0,0) and moved 2 steps up, it would be at (0,2) and all pixels along the line from (0,0) through (0,2) would be shaded; 3 pixels in total.)

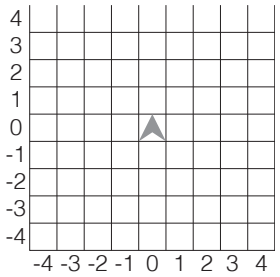
For (a), (b), (c), and (d) **we start with the pen down, the sprite in the middle of the grid, facing up,** as shown. Your job is to shade in (completely!) *all* the pixels that will be colored in after 

.

…and here is how we would have written each if we were texting our friend our code (note the indenting):

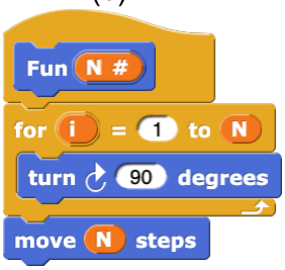
(a)

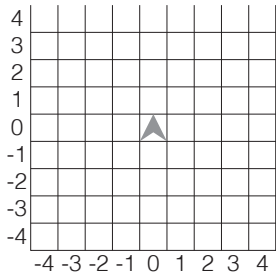





```
Fun(N)
for i = 1 to N
--> move 1 steps
turn right 90
```

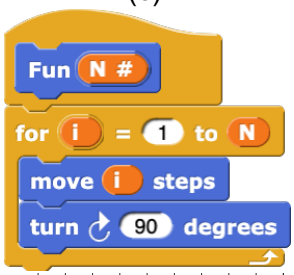
(b)

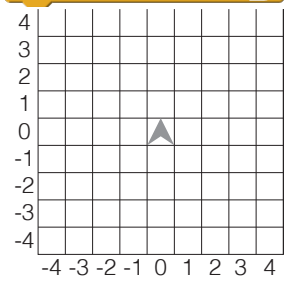





```
Fun(N)
for i = 1 to N
--> turn right 90
move N steps
```

(c)






```
Fun(N)
for i = 1 to N
--> move i steps
--> turn right 90
```

d) Well, all of those were pitiful attempts at writing code that would have the sprite spiral outward perfectly, like the picture below. If `N` were big enough, `Fun(N)` would eventually shade every pixel.
Write the code for `Fun(N)` that does this in the lines below, using the "text your friend" style we show above. Make sure to use arrows to indent the inside part of any `for` loop you use. You might not need all the lines.

```
Fun(N)
```

_____

_____

_____

_____

_____

_____