University of California, Berkeley – College of Engineering
Department of Electrical Engineering and Computer Sciences
Fall 2010 Instructors: Dan Garcia and Brian Harvey 2010-10-27

CS10 Paper Midterm

Last Name	
First Name	
Student ID Number	
`cs10-` *Login First Letter*	**a b c d e f g h i j k l m**
`cs10-` *Login Last Letter*	**a b c d e f g h i j k l m** **n o p q r s t u v w x y z**
The name of your LAB TA (please circle)	**Jon Luke**
Name of the person to your Left	
Name of the person to your Right	
All my work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS10 who have not taken it yet. ***(please sign)***	

Instructions

- Don't Panic!
- This booklet contains 6 pages including this cover page. Put all answers on these pages; don't hand in any stray pieces of paper.
- Please turn off all pagers, cell phones and beepers. Remove all hats and headphones.
- **Question 0 (1 point) involves filling in the front of this page and putting your login on the top of every sheet of paper.**
- You have 110 minutes to complete this exam. The midterm is closed book, no computers, no PDAs, no cell phones, no calculators, but you are allowed two double-sided sets of notes. There may be partial credit for incomplete answers; write as much of the solution as you can. When we provide a blank, please fit your answer within the space provided.

Question	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Online	Total
Points	1	3	2	2	3	3	3	3	2	2	3	5	10	8	10	60

Short-answer Questions

Question 1 : A “sea-change” in CPU design occurred around 2004 and most agreed the “free lunch” was over. *In one sentence*, what caused it and what did manufacturers do in response?

> In the never-ending quest for increasing CPU performance [1 pt], chips running at high clock frequencies were getting too hard to cool (the power density was approaching that of a nuclear reactor) [1 pt] so they decided to put multiple, slower, cooler cores on a chip [1 pt].

Question 2: You have a piece of code, half of which is executed serially, the other half is executed in parallel. In the ideal case with an infinite number of “helpers” dividing up the parallel portion, what is the overall speedup of your code? _____

> Amdahl’s law states that the speedup in the perfect, infinite-helpers case is 1/s, so the speedup is 1/0.5 = 2x, or twice as fast. People received full credit for writing 1/2, because we assumed they were talking about the time (inversely related to speedup). Rubric was pretty much 2 or 0.

Question 3: Let us imagine Twitter mandated that every one of their users “follow” every other user. How does the *total* number of “connections” between people (connections were visualized as arrows, or links in the graph) scale with the number of users?

(a) It remains relatively constant, regardless of the number of people.
(b) It scales linearly with the number of people.
(c) It scales quadratically with the number of people.
(d) It scales exponentially with the number of people.

> If every one of Twitter’s N users followed every other (N-1) user, that would be N * (N-1) = N^2 - N connections, which is (c) quadratic growth.

Question 4: *In one sentence*, how are search engines (e.g., Google) able to have a tweet available in their search results in less than a second after the tweet happens? (It normally takes a search engine hours to “crawl” the web and update their search index.)

> Twitter “pushes” the results to Google (and Google pays handily for the data). Credit given according to how close you were to this idea. Describing Google’s crawling & indexing process is not an appropriate response for this question.

Question 5: Fill in the blanks with the appropriate technology that has changed the world.

a) _____ is revolutionizing the way people collaboratively author documents.

b) _____ and _____ are a hardware and software combination that put professional-quality document production in the hands of the masses.

c) The Internet was “invented” in 1962, but it wasn’t until 1993 when Marc Andreesen at NCSA introduced _____ (known as its first “Killer App”) did its use explode.

a) Google Docs (and other Software as a Service (SaaS) utilities), Wikipedia too.
b) The laserwriter and Postscript. The phrase “put professional-quality in the hands of the masses” was meant to imply that it was one of the first to do it. So WYSIWYG and Microsoft Office are half-right (but they’re both software). They too made life easier but without the laserwriter, you don’t get professional-quality output by your desktop.
c) his web browser, NCSA Mosaic.
Grading standard: of the 4 blanks,
4 right = 3 pts.
2-or-3 right = 2 pts.
0-or-1 right = 0-or-1 pts.

Question 6: Give an argument why educational microworlds might be preferable to either pure tools or pure courseware.

Courseware devalues creativity and critical thinking in favor of memorization, but tools make it too easy for a learner to get lost down dead ends or not know where to begin. Microworlds establish the general topic of exploration but don't pose overly specific problems.

Scoring: 3 -- correct.
2 -- discusses just tools, or just courseware.
1 -- incoherent answer, but some glimmer evident.
0 -- other.

Question 7: Besides obeying local laws, name one reason why Google makes deliberate (non-automatic) interventions in the display of search results.

My answers:
1. Removing a result at the request of its owner (e.g. for copyright reasons).
2. To prevent or punish attempts to mislead the ranking algorithm.
3. Special arrangements with high-volume sources (e.g. Twitter, CNN).

The answer "to sell advertising" is not really right; Google does display ads next to the search results, but they aren't part of the display of the results. But we accepted it if there was enough detail to make it clear that they don't (unlike some other search engines) reorder results

that aren't marked as ads for money.

Scoring:

3. correct.
2. answer cites local laws, e.g., in China or for pornography in the US.
1. true but irrelevant statements about how Google displays searches.
1. many answers given, some of them okay.
0. just describes PageRank (which is automatic, not manual).

Question 8: Why did the writers of the US Constitution require a limit to the duration of patents?

My answers:

1. So inventions don't die with their inventors.
2. Patent is a bargain between society and inventors, not a right.
3. So other inventors can build on this invention.

Scoring:

2. Any of those.
1. I couldn't quite decide.
0. Maybe the patented invention becomes invalid.
   (Why after a fixed time? In reality, a patent might be ruled invalid for some reason, in which case it's retroactively invalid from the beginning. Or a patent may become *irrelevant* because of later inventions, so it isn't worth money any more, but it's still valid.)

Login: **`cs10-____`**

Question 9: The Giants win the Pennant! The Giants win the Pennant!

Provide a single, simple expression that reports **`SF`** from the nested list **`playoffs`**. On the right is a screen capture of the Variables tab if that helps.

Answer:

You received the full **2** points if you answered (item (1) of (item (3) of (playoffs))) or provided an equivalent answer. You received 1 point if you used incorrect indices or nested the "item" blocks incorrectly; for example, (item (item (3) of (playoffs)) of playoffs) -- this is incorrect because the result of (item () of ()) is a list in this question, and you cannot put a list into an input meant for a number. You also received 1 point if you returned a list containing "SF", instead of "SF" itself; this is, for example, the result of simply (item (3) of (playoffs)). All other incorrect answers received 0 points.

Question 10: Faster, Puppydog! Live! Live!

You wish to fill a list **`nums`** with the numbers from 1-100 (you don't care about order). So, you write the following (the **`for`** block simply goes through like a **`repeat`** block with a built-in index, called **`i`** in this case):

which works perfectly, but it's a little slow. However, your friend asks *"why don't you break the problem into two parts and fill the list in parallel!"* So, you duplicate that script, and adjust the numbers in the **`for`** loop like so:

In the best case, this works! What happens (or could happen) in the worst case and why?

> The list would have fewer than 100 items because of the concurrency *race condition* on the multiple "**`delete all of nums`**" commands (this is actually what happens in Scratch/BYOB). The 51 never makes it to the final list because the right script finishes its "**`delete all of nums`**" then "**`add 51 to nums`**" first, THEN the left script executes its "**`delete all of nums`**" which *removes* the 51. In the absolute worst degenerate case, one block could add all 50 of its numbers only to have them deleted by the other block ... yikes!
>
> You received the full **3** points for a correct answer, or if you mentioned that there would be a "race condition", where less than 100 numbers would be present in the final list, or where some numbers would be deleted. If your answer mentioned that one script would overwrite the result of the other script, or if your answer does not mention that elements from the final list would be deleted, or if your answer lacked mentions of a "race condition" but implied the existence of one, then your answer received 2 points. If your answer mentioned that only one script would run, or that there would be multiple errors but a correct final result, it received 1 point. All other incorrect answers received 0 points.

Question 11: Talking' about my gggggeneration...

If you've ever spilled soda into a keyboard, you know that keys start to stick and repeat. We want to write a block to remove the extra consecutive *repeat character* we request, leaving only one. E.g., if the sentence were:

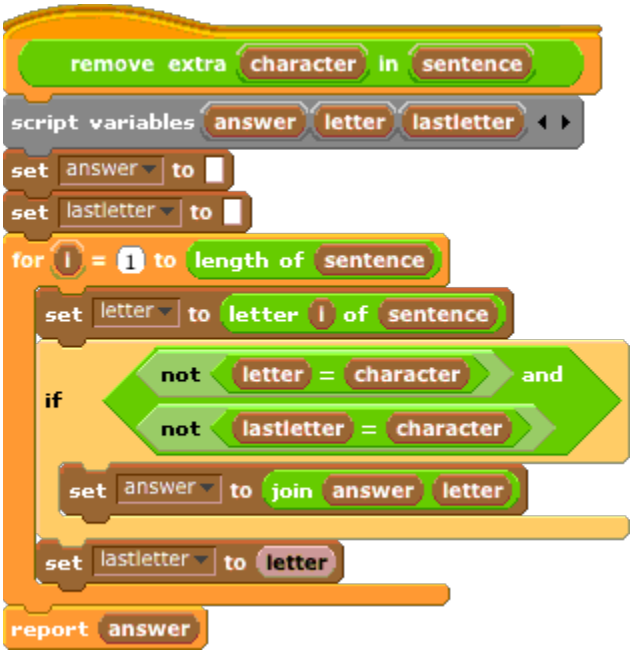
`I love CS10..... Me too..... Me three!!!`

...and we asked our block to remove the extra "`.`"s, we would get:

`I love CS10. Me too. Me three!!!`

(We'll limit this to punctuation, since the meaning could be lost if we used it on letters and numbers. Imagine: "**`The national debt is $10000000000000!!`**" would become "**`The national debt is $10!!`**")

We've tried to write code to do this for us, but we believe it has a bug. Briefly, **`answer`** and **`lastletter`** initialize to the empty character, and then the **`for`** loop (these were explained in question 10) sets **`letter`** to every character as we walk forward in the **`sentence`**. We only add that **`letter`** to the **`answer`** (which we report at the end) if the **`letter`** is not the **`character`** to remove and not the same as the previous character **`lastletter`** (to prevent the repeats).



a) Describe all **`sentence`**s that don't trigger the bug, i.e., our program returns a correct answer.

All **`sentence`**s that do not have the input character.

b) Briefly describe the small change(s) needed to fix the bug.

Change the **and** to an **or**.

The **5** points for this question were split between **3** points for part (a) and **2** points for part (b).

Your answer for part (a) received **1** point if it provided a subset of the correct answer: for example, many answers claimed that the code would only work "if the first letter is not `character`". While this is true, it does not include all `sentence`s that do not trigger the bug. If you gave specific examples, then your answer received no points.

Your answer for part (b) was graded based on your answer for part (a): if your suggested change correctly fixed the bugs in the sentences that you described in part (a), you received 2 points even if your answer to part (a) was incorrect, but such answers were rare. In general, many students correctly answered part (a), but suggested a fix that would not work for all input `sentence`s: such answers generally received 1 point.

For example, a common, but incorrect, suggested fix was to remove the `if`-constraint that `letter` should not be equal to `char`; in other words, the script should only add the current letter to the final answer if the last letter does not match the character. This would correctly replace consecutive repeated characters with one of the characters, but it would not include the letter that comes just after a consecutive string of these repeated characters. For instance, in the sentence `abbbbcd`, the `b`'s would be reduced to one `b`, but the `c` after the `b`'s would not be included, resulting in the sentence `abd`. Another common, but incorrect, suggested fix was to replace the `if`-constraint with the constraint `(not (last-letter = letter))`; in other words, the script should only add the current letter to the final answer if the last letter is not equal to the current letter. This would definitely work and replace all consecutive characters with one character. However, this means that it would replace *all* consecutive characters, even the ones that are different from the `character` provided to the block. Thus, the sentence `abbbcddde` would result in `abcde`, even if we specify that the only character we need to consider is `b`.

A few students suggested the constraint `(not (letter = character)) or ((letter = character) and (not (lastletter = character)))`, which basically says that the current letter must be added to the answer if it is not `character`, but if the letter *is* `character`, then we must ensure that the last letter is not `character`. This is the equivalent to our fix, but is perhaps more intuitive. This answer received 2 points.

Login: **cs10-____**

Question 12: Two roads diverged in a wood...

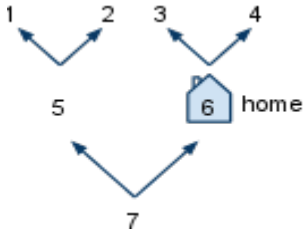
You're lost in the forest. Every (place) in the forest is either a *dead-end* or has exactly 2 *one-way paths*: *left* and *right*. Your goal is to find out if there is a way home. We introduce a new data type called a (place), but you don't know *(and you don't need to know)* how it is represented; it could be a string, a number, or a list. You are presented with four new blocks, two predicates and two reporter blocks (all take a place as an argument):

- (home? place) returns (true) if the (place) is your home, (false) otherwise.
- (dead-end? place) returns (true) is the place is a dead-end (i.e., no paths from it).
- (go-left place) follows the *left* path, returning a new (place).
- (go-right place) follows the *right* path, returning a new (place).

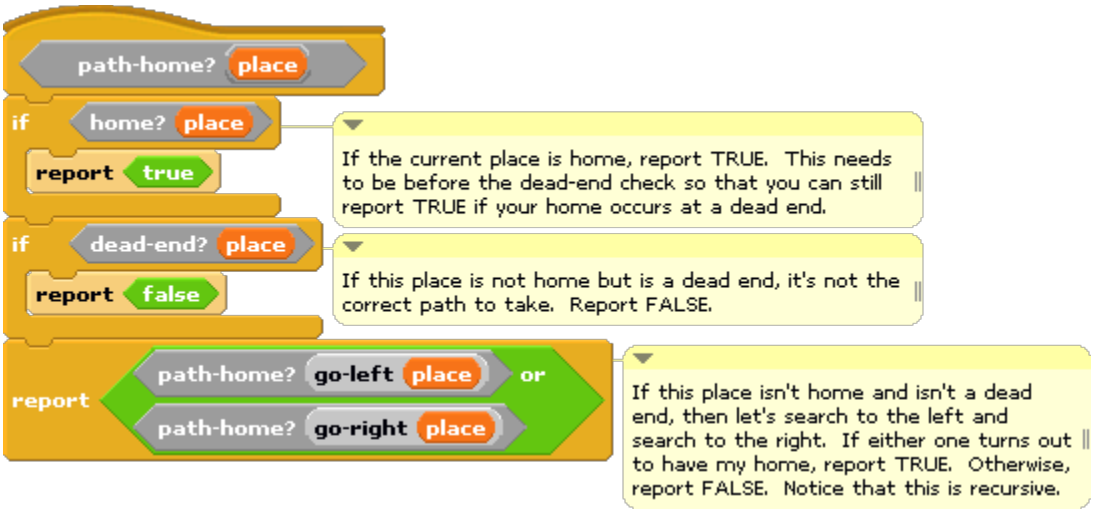
It is an error to (go-left place) or (go-right place) if (place) is a *dead-end* (because it has no paths!). There is no way in this forest to follow a sequence of left paths and/or right paths and end up where you started. I.e., there's no way to walk in circles. Your *home* (if one exists) might be at a dead-end or it might not. You might actually start your search at home.

Write (path-home? place), which uses the four functions above and returns (true) if you can get home following a (possibly zero) number of lefts and rights starting from (place), and (false) otherwise. Use the technique we described for authoring BYOB code on paper. We've provided an example forest for you, but **your solution needs to be able to work with ANY forest.** Below, we present a table that shows the responses of various blocks when you are at different places in the sample forest on the lower right.

place	home? place	dead-end? place	go-left place	go-right place	path-home? place
1	false	true	ERROR	ERROR	false
2	false	true	ERROR	ERROR	false
3	false	true	ERROR	ERROR	false
4	false	true	ERROR	ERROR	false
5	false	false	1	2	false
6	true	false	3	4	true
7	false	false	5	6	true



Answer:



The ten points for this problem were distributed for five different parts of the problem.

(2 pts) Did you check whether the current location was home for the first half of the base case? You had to do this before checking for a dead-end.

(2 pts) Did you check whether the current location was a dead-end? You had to do this after checking to see if it was your home.

(2 pts) Did your code search down the right path if it didn't find home or a dead-end at the current location?

(2 pts) Did your code search down the left path if it didn't find home or a dead-end at the current location?

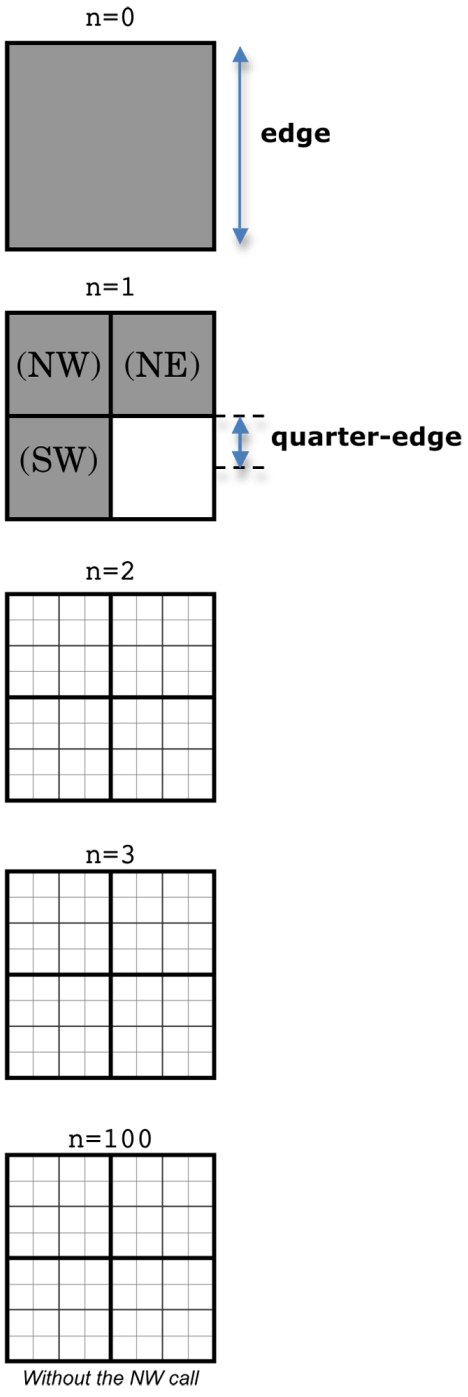
(2 pts) Did you get all `report` logic correct, and does your code work in all cases?

Partial credit was also given in particular categories if you were close to the correct answer but still missed something somewhat important.

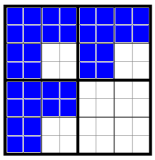
Question 13: Anyone up for a game of checkers? Hey, you, Sierpinski!

We've designed a fractal "checker" whose base (n=0) case simply stamps out a filled square sprite (initially 360x360 pixels), and whose recursive case places three half-sized copies of the previous generation in the Northwest (NW), Northeast (NE) and Southwest (SW) corners. The code that generated them is on the left, **but you can answer the question without it.**

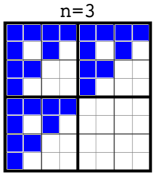
a. Sketch the **`n=2`** fractal below on the right.
b. Sketch the **`n=3`** fractal below on the right.
c. If we forgot to write the Northwest (NW) recursive call (the middle call to **`checker`**), roughly sketch what the **`n=100`** call would be.



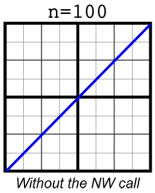
n=2



n=3



n=100



Without the NW call

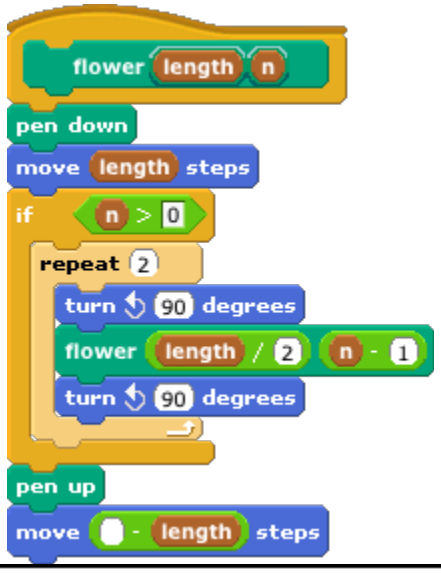
The **8** points for this problem were split between **5** points (in total) for the fractals where n = 2 and n = 3, and **3** points for the question that asked what the n = 100 call would look like, in the absence of the NW recursive call.

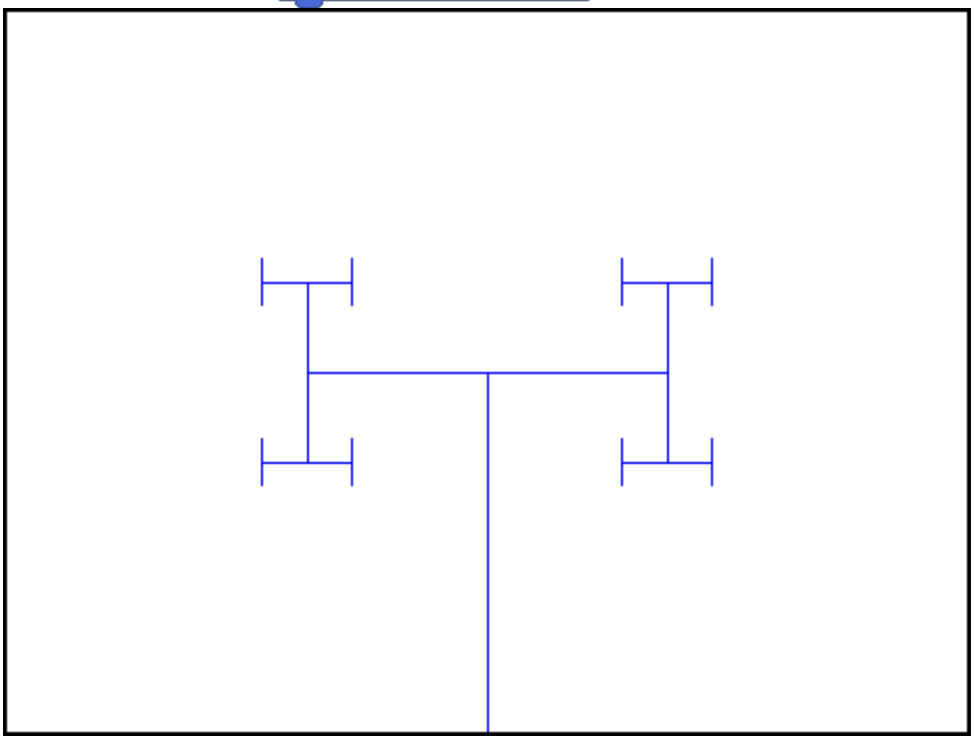
You received 3 points (out of 5) for the first two parts if your answers were incorrect but *consistent*: it looked like your answers for the two fractals were related, and the fractal for n = 3 could conceivably be obtained from the fractal for n = 2 using appropriate recursive calls.

You received 1 point for the last part if you had a “checkerboard” pattern, which possibly implied that you did ignore the NW recursive call, but still called the resulting pattern recursively in all four directions. All other incorrect answers received 0 points. The most common mistake on this part was to forget that ignoring the NW recursive call also implies that you never go into that region of the fractal “checker”.

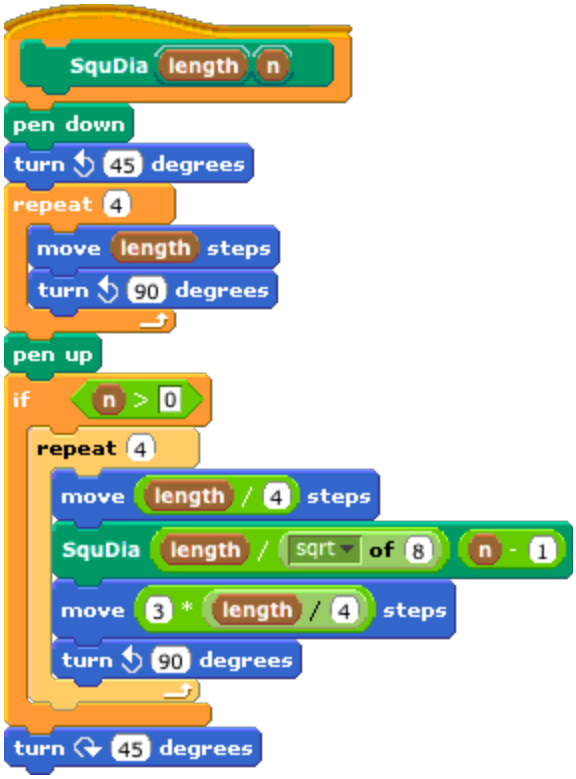
2010Fa CS10 Online Midterm Answers

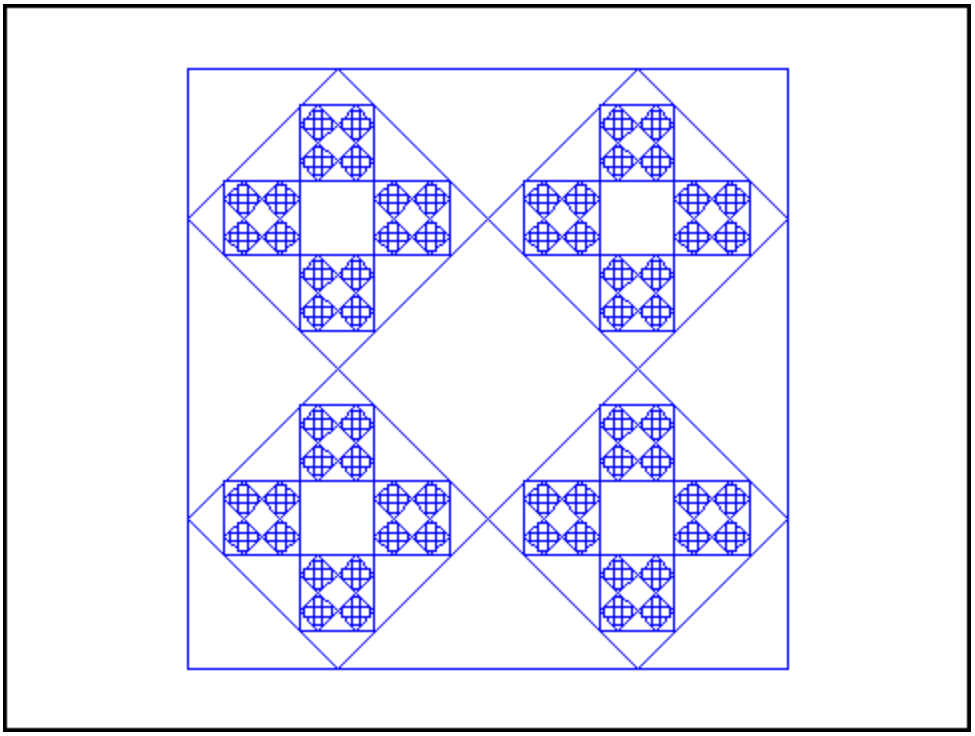
Fractal Tree:





Fractal Square Diamond:





Fractal Sierpinski Triangle: