

CS10 Midterm

<i>Last Name</i>	
<i>First Name</i>	
<i>Student ID Number</i>	
<i>The name of your LAB TA (if you remember)</i>	
<i>Name of the person to your Left</i>	
<i>Name of the person to your Right</i>	
<i>I have neither given nor received any assistance in the taking of this exam. (please sign)</i>	

Instructions

- **Don't open this test until you are told to do so!**
- **Remove the last sheet of paper from the exam, which you should use as a reference.**
- This booklet contains 8 pages including this cover page and the two supplementary back pages. Make sure to remove the last sheet of the booklet.
- The test contains 6 pages. Put all answers on these pages; don't hand in the supplementary sheet or any stray pieces of paper. Answers written somewhere other than pages 1-6 will not be graded.
- After you start, **make sure to write your SID on the top of every page of the test!**
- The in-lab portion of this midterm was 15 points. This exam covers the remaining 60.
- Please turn off all mobile phones or things that make noise. Remove all hats and headphones.
- You have 110 minutes to complete this exam. The Midterm is closed book, no computers, no cell phones, no calculators, but you are allowed two double-sided sheets with handwritten 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.
- The first 11 problems cover lecture/reading material.
- Problem 14 is particularly difficult.
- Don't Panic!

Question	1-11	12	13	14	Total
Points	20	17	8	15	60

SID (write on every page): _____

Q1-11: Reading/Lecture Questions (2 pts each)

Out of problems 1 -11, we will drop the lowest-scoring question

Question 1: In Gerald's Privacy talk, we discussed the dangers of how latitude/longitude information (a.k.a. geotagging) embedded in photos and videos could be used to allow thieves to collect information useful for theft (e.g. that you are on vacation, where you live, etc.) Gerald explained that simply removing/outlawing geotagging information from photos and videos would not really help in the long run. Why?

Your answer:

Question 2: At the end of Valkyrie's talk on human computer interaction, we saw a bunch of neat research projects featuring devices with novel user interfaces. Give a short description of one of these devices:

Your answer:

Question 3: Until roughly 2002, the frequency of computer processors increased dramatically with each passing year. Why did this process stop? Circle one:

- a. Transistors stopped getting smaller.
- b. Computer chips stopped getting bigger.
- c. Computer chips started giving off too much heat.
- d. Chip manufacturers started putting multiple processors on a single chip.

Question 4: In lab 11, the lab text says: "Let's try to use concurrency for what it was meant for: **speed!**" I disagree with the idea that concurrency is primarily meant for speed (but forgot to edit this line of the lab, sorry). Give another example of why concurrency is useful other than speed.

Your answer:

Question 5: In "As We May Think" by Vannevar Bush, the author describes something called the "memex", "in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility". What is the big difference between the "memex" as envisioned by Vannevar Bush and the world wide web (Hint: Valkyrie mentioned this difference in class)?

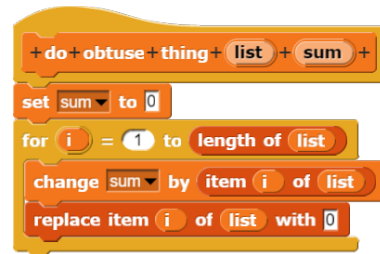
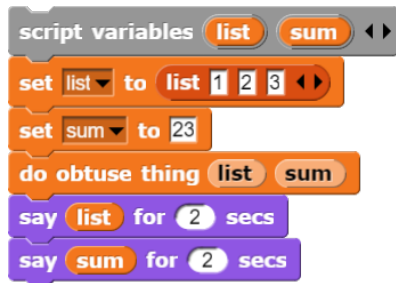
Your answer:

Question 6: In Blown to Bits, Chapter 2, the author mentions that even if you wear gloves, someone could still possibly track you down if you carefully send an anonymous laser printed note (going so far as to wearing gloves while handling the paper). What technique does he suggest?

Your answer:

SID (write on every page): _____

Question 7: Suppose we run the script on the left, with block definition on the right.



What will the “say list” command say?

Your answer:

Question 8: What will the “say sum” command from question 7 say?

Your answer:

Question 9: What is the primary reason that we are moving from IPv4 to IPv6 (i.e. from 32 bit IP addresses to 128 bit IP addresses)?

Your answer:

Question 10: When I email my dad (who uses sbcglobal.net for email), I send packets from my computer to gmail, and gmail then sends along packets representing my email to sbcglobal. Why is this scheme better than sending the packets directly from my computer to my dad’s computer?

Your answer:

Question 11: Suppose we have an algorithm that operates on lists, and which has a runtime that is quadratic as a function of the length of the list. Suppose we create an implementation of this same algorithm that breaks the problem into 8 pieces, each of which is run on a separate CPU on a computer with 8 CPUs. What will be the new order of growth for the runtime? Circle one:

Circle one: Constant, Logarithmic, Linear, Quadratic, Cubic, Exponential, Worse Than Exponential

SID (write on every page): _____

Question 12: Recursive Reporters (17 pts)

a) (3 pts) A CS10 student working on the recursive reporters lab implements the Odd Numbered Items Of reporter in a mutually recursive way, as shown below:



What does **odd numbered items of** `list 11 12 13 31 34 35` report? Give your answer in the blank below. If the reporter doesn't report anything because it gets caught in an infinite loop, write "infinite loop".

b) (7 pts) Write a **recursive reporter** block "count up" that reports a text equal to all values between lo and hi. You cannot use any loops (for, while, repeat, etc.) in your code. As an example:



You may assume hi is greater than or equal to lo. You may also assume that lo and hi are zero or larger. Place your answer in the blanks below. Make sure to use the "report" block. **You must use the code writing technique described on the last page of this exam.**

_____ count-up(LO, HI): _____
_____ if _____
_____ else _____

SID (write on every page): _____

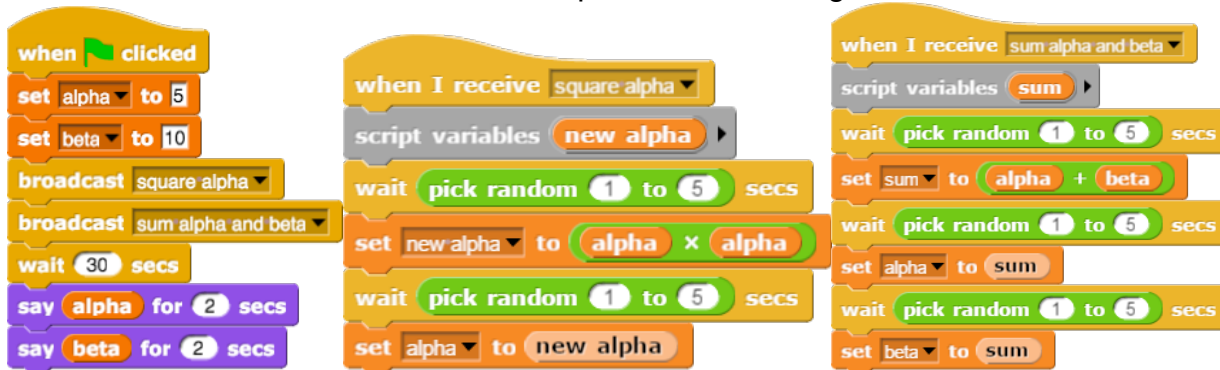
c) (7 pts) Write a **recursive reporter** “all count ups” that reports a text equal to all counts up between 1 and n. You cannot use any loops (for, repeat, repeat until etc.) in your code. You may use the “count up” block from part b in your code. For example:



_____all-count-ups(N):_____

Question 13: Concurrency (8 pts)

Consider the concurrent code below¹. Alpha and beta are global variables.



After clicking the green flag, what are all of the possible values that might be said for “alpha”? What are all of the possible values that might be said for “beta”? You may not need all of the blanks.

Possible alpha values:



Possible beta values:

¹ If you’re wondering why we always do this silly thing where we “wait” in code for no clear reason: When you run code in most programming languages, there is effectively a very tiny random wait after each statement in the script is executed. In CS10, we manually add these random waits to increase the chance that we observe a race condition.

SID (write on every page): _____

Question 14: The Law of the Broken Futon (15 pts)

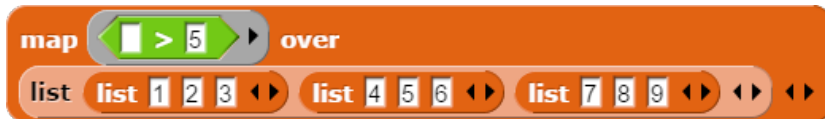
For each piece of code below, write the corresponding answer. **Give your answers using the notation on the removed page.** To answer these questions, you'll need two key facts that we haven't learned before.

Fact 1: Any arithmetic reporter that receives a non-numerical input (for either or both inputs) will report the text "NaN", meaning "not a number". For example:  NaN and  NaN

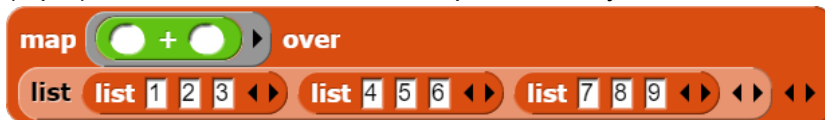
Fact 2: The predicate  considers **any list greater than any number**, e.g.

 false  true There is no good reason that this predicate should behave this way, and comparing a list with a number is not something you should ever do when programming. For this problem, simply accept that this is Snap!'s weird way of handling a weird situation.

a) (3 pts) What will the code below report? Write your answer in the blank below the code.



b) (4 pts) What will the code below report? Write your answer in the blank below the code.



c) (4 pts) What will the code below report? Write your answer in the blank below the code.²



d) (4 pts) What will the code below report? Write your answer in the blank below the code.

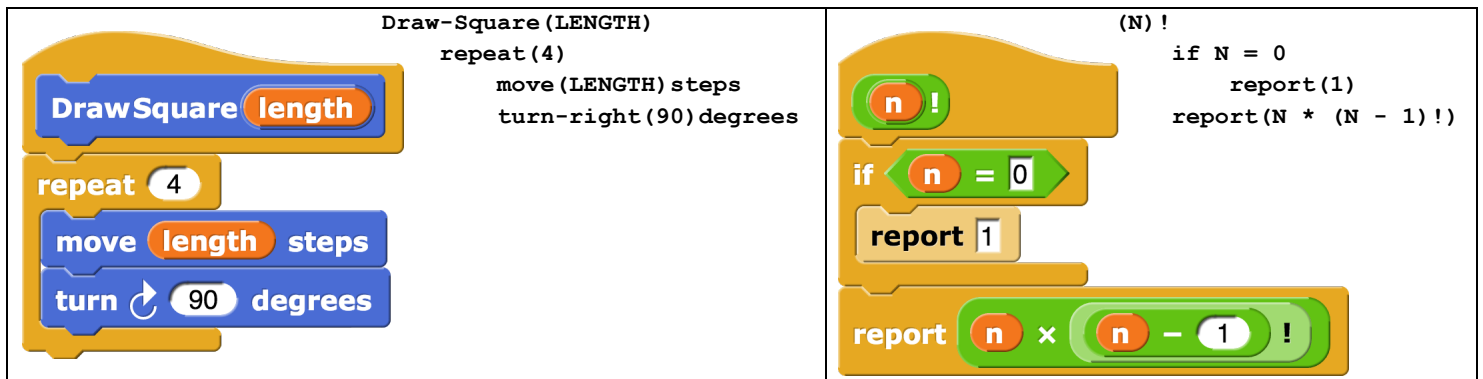


² Hint on part c:  is a reporter with ONE blank input, not three. If this hint is confusing, ignore it.

Writing *Snap!* code on paper (supplementary)

You will be asked to write *Snap!* code on this exam, so we've developed a technique for writing it on paper. There are a few key things to notice:

- We often write variables in **UPPERCASE**.
- We change spaces between words in block names to dashes (this makes it much easier to read).
- We use indentation just as *Snap!* does, to help us understand what is "inside" the **if**, **else**, and other Control structures. E.g., here's how you could write the **DrawSquare** and **n!** blocks:



- When you want to write a list of things, write them with an open parenthesis, then the first item, second item, etc (separated by spaces) and when you're done, put a closed parenthesis. If any of your items are a sentence, you have to put quotes around the sentence. So, for example, the following list of three things would be written as the equivalent 3-element-list:

■ (life liberty "pursuit of happiness").



- Similarly, a nested list just shows up as a nested set of parenthesis. So the following would be written as

■ ((Love 5) (Hate 4) (The 10)).

- If you want to pass in a function as argument, you know the function must be surrounded by a grey-border. Here are three new conventions:
- The grey border is written as *square brackets*: []
- Blanks are written as parenthesis with underscore `_` in the middle, but common blocks that are passed in to HOFs can be simplified by just their name (and not the parens and underscores)

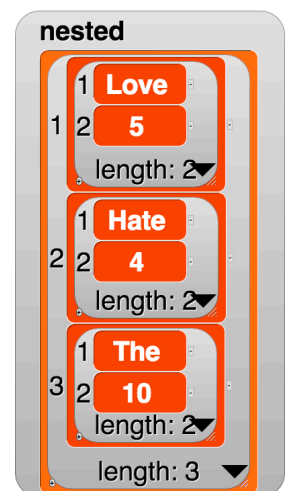
■ Return values are written as \rightarrow value

- So the following would be written as:

■ Map[(_) * (_)] Reduce[(_) + (_)] over ((1 20 3 10)) \rightarrow 510

- or, in the more simplified (and preferred) format:

■ Map[*] Reduce[+] over ((1 20 3 10)) \rightarrow 510



A bunch of Snap blocks are shown below as a reference. For coding problems on this exam, you may use any Snap! block, not just the ones below (we've omitted lots of them, like +, -, split, etc.)

