SID: ________________________________

University of California, Berkeley – College of Engineering
Department of Electrical Engineering and Computer Sciences
Fall 2014 Instructor: Gerald Friedland 2014-12-16

CS10 PAPER FINAL

Last Name		
First Name		
Student ID Number		
The name of your LAB TA (please circle)	**Adam Andy Arany Jaclyn Janna Joseph Jeffrey Max Rachel Sumer Steven Victoria**	
Name of the person to your: Left | 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 haven't taken it yet. ***(please sign)***		

Instructions

- Don't Panic!
- This booklet contains 7 (double-sided) pages including this cover page.
- Please turn off all pagers, cell phones and beepers. Remove all hats and headphones.
- You have 170 minutes to complete this exam. The exam is closed book, no computers, no PDAs, no cell phones, no calculators, but you are allowed **three** double-sided sets of notes.
- There may be partial credit for incomplete answers; write as much of the single, correct solution as you can.
- Please fit all answers in the boxes provided! Don't hand in any extra paper!

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**Online**	**Subtotal**	**1-11 low**	**Total**
Points	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**12**	**18**	**6**	**9**	**20**	**109**	**4**	**105**

If you can draw and you have time, feel free to doodle all over this front page!

SID: ______________________________

(Really) Short Answer Questions (*4 points each, lowest dropped, 45 min*)

WRITE YOUR ANSWERS WITHIN THE BOXES. Anything outside the boxes WILL NOT be graded.

Question 1: Name two of the principles of online privacy discussed in lecture. For **each** principle briefly explain what you can do to protect yourself.

Question 2: What is the difference between qubits and bits that makes quantum computing both faster and more challenging than regular computing? Why?

Question 3: In recent years, there has been much talk about the data explosion, particularly about big data and data mining. What is the challenge with big data? What is data mining, and why can it be said that data mining is an iterative process?

Question 4: APIs are often described as black boxes. What does API stand for? What does it mean to be a black box and why is this helpful when programming?

Question 5: Ivan Sutherland, Turing Award winner, is well known for creating *Sketchpad*. What did *Sketchpad* do and why was it so revolutionary? What are the two programming paradigms shown his invention?

Paradigm 1: ______________________ Paradigm 2: __________________________

Question 6: From Alyosha Efros' lecture, what is computer vision? Name a problem that computer vision has helped solve.

SID: ______________________________

Question 7a: Which best describes the *knapsack problem* from lecture? (**CLEARLY** circle one)

(a) Tractable-with-a-polynomial-time-solution (b) Intractable-optimally-*but-has*-a-reasonable-time-approximation

(c) Intractable-optimally-*and-doesn't-have*-a-reasonable-time-approximation-solution (d) not-solvable

Question 7b: The halting problem was one of the first examples of what type of problem? What question does this problem try to answer? Was this problem solvable and why?

Question 8: In 2004, there was a 'sea change' to multi-core parallelism for computer architects. **CLEARLY** circle all the things that it forced the computing community to rethink.

Languages Architectures Algorithms Data Structures

Question 9: According to Obama, name and explain what he considers to have been 'built into the fabric of the Internet since its creation'. Also, name one of the three factors that drove the design of Obama's proposed rules about this.

Question 10: Fill in the Blank -- From Blown to Bits: 'To *regulate speech* on the Internet, lawmakers can target the source, the destination, or ___________ '

Question 11: According to Eric Paulos, Steve Jobs described HCI as the intersection of what two fields? Explain the importance of this intersection to HCI.

SID: ______________________________

Question 12: ***Snakes on a Test!*** *(12 points, 24 minutes)*

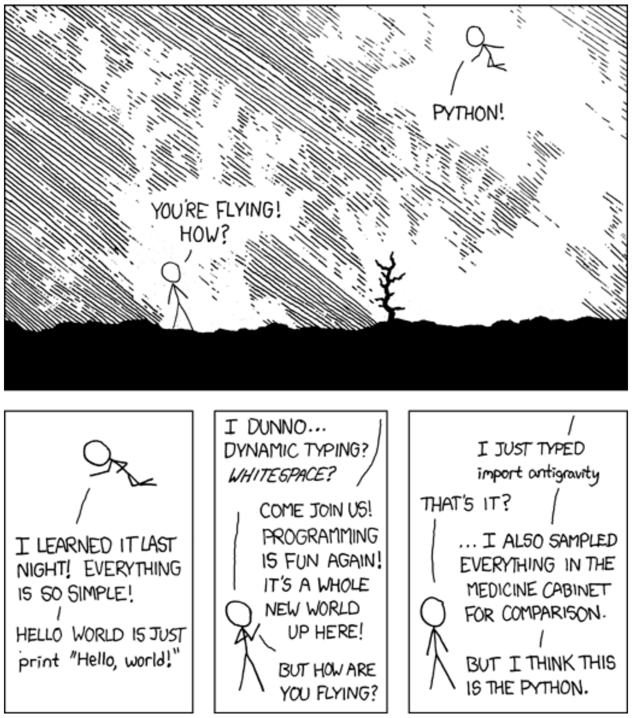
Part A: What are the results of the following Python expressions? Write the results in the correct form as numbers, strings, or lists in the corresponding boxes.

For Example:

`0) [1] + [2] =`	[1,2]
`a) [ ] + [1,2,3] =`	a)
`b) ‘race’ + ‘car’ =`	b)
`c) [1, 3, 5, 2, 4][2:4] =`	c)
`d) ‘1234567890’[-3:] =`	d)
`e) [[‘I’, ‘love’], [‘CS’, ‘10’]][0][1] =`	e)
`f) {‘a’ : 1, ‘b’: 2, ‘c’: 3}[‘b’] =`	f)
`g) {‘c’ : 3, ‘d’: 4}.keys() =`	g)

Try this at home in the Python interpreter!

```
import antigravity
```


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

SID: ______________________________

Part B: Here is a function, `copy()`, that is meant to take in a list and return a copy of that list. However, there are bugs in this code! Find as many as you can, and write a corrected version below.

```
def copy(lst):
    new_list == []
    i == 1
    while i < len(lst):
        new_list.append(lst[i])
    return new_list
```

Example (assuming the function works correctly):

```
>>> copy([4, 8, 2, ‘hello’, 9.0])
[4, 8, 2, ‘hello’, 9.0]
>>> a = [4, 8, 2, ‘hello’, 9.0]
>>> b = copy(a)
>>> b
[4, 8, 2, ‘hello’, 9.0]
```

We’ve found 4 bugs in the given code, and there are many different ways to fix these bugs. However, please use the same variable names and do **not** substitute the `while` loop for any other loop (i.e. `for` loops are not allowed). You may change the condition of the `while` loop and the code inside the loop. You may not need all the lines.

SID: ______________________________

Question 13: Drawing Algae and Dust *(18 points, 36 minutes)*

***Note**: This question has multiple parts, and some are independent! Please feel free to solve these out of order. If you get stuck, move onto parts E and F.*

Astrid Lindenmayer came up with a compact way of capturing growth patterns into a series of *rewriting rules* that lent itself to describing fractals quite efficiently; they came known as 'L-systems.'

The original L-system modeled the growth of algae as follows:

1. Start with the letter **A**
2. For every subsequent generation, simultaneously replace (from the previous generation) every…
    - **A → AB**
    - **B → A**

Here are the first few generations of the algae growth pattern:

N = 0: **A**	Initial generation.
N = 1: **AB**	A spawned into AB by rule (A → B). Note: Rule (B → A) could not be applied.
N = 2: **ABA**	A spawned into AB again, former B turned into A.
N = 3: **ABAAB**	The breakdown from step 2 → 3 looks like this: **A | B | A → AB | A | AB**

a) What is the N = 4 generation?

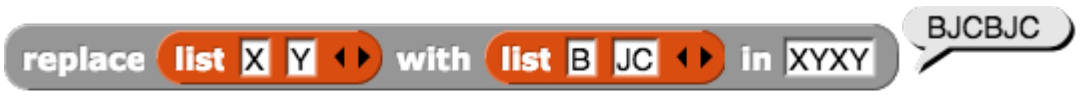
b) The length of the growth pattern follows what very famous integer sequence? (If you need to, write out levels 5 or 6.)

SID: ______________________________

c) Fill in the blank in `algae` so that `algae(N)` returns the Nth generation. You will find the function `replace (in) with (out) in (sequence)` quite handy. We've given an example of the `replace` block after this code – take a look at it before solving this problem.

```
algae(N)
     if N = 0
          report A
     else
          report ( ________________________________________________ )
```

The **`replace`** block takes in two lists (**`in`** and **`out`**) and a character sequence (**`sequence`**). Each character in **`sequence`** can be found in the list **`in`**. That character is replaced by the corresponding value in the second list **`out`**. Here is an example to show the **`replace`** function usage. Each instance of X is replaced with B and each instance of Y is replaced by JC.



```
replace list X Y with list B JC in XYXY
BJCBJC
```

Cantor Dust is a neat fractal which can also be modeled as an L-System. Here's what it looks like:

Like algae, the rules for Cantor Dust are simple.

1. Start with the letter **A**
2. For every subsequent generation, simultaneously replace (from the previous generation) every…
   - **A → ABA**
   - **B → BBB**

Here are the first few generations of the Cantor Dust pattern:
N = 0: **A** Initial generation.
N = 1: **ABA**
N = 2: **ABABBBABA**

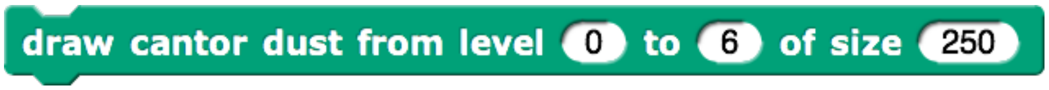
d) Describe what (simple) change you can make to the block `algae(N)` to make a new function: `cantor dust (N).`

SID: ______________________________

e) The following question can be solved independently from the rest of Question 13.

One super-awesome thing about L-systems is that they're easy to turn into real drawings. Below is the code you'll need to draw the Cantor Dust fractal. However, we've left some code for you. The block draw `cantor dust from (start) to (stop) of size (size)` will draw the given rows of the fractal.

Here's the block we used to draw the picture above. If it helps, you can assume the sprite started from the top left corner and was facing to the right. The top row is 0, and the bottom row is row 6.



```
draw cantor dust from level (0) to (6) of size (250)
```

This block will draw one row of the Cantor Dust image at a time. It uses **`cantor dust (n)`** block, from Part D, which will return a sequence of letters like: '**ABABBBABA**.' You may assume the block works correctly for this part.

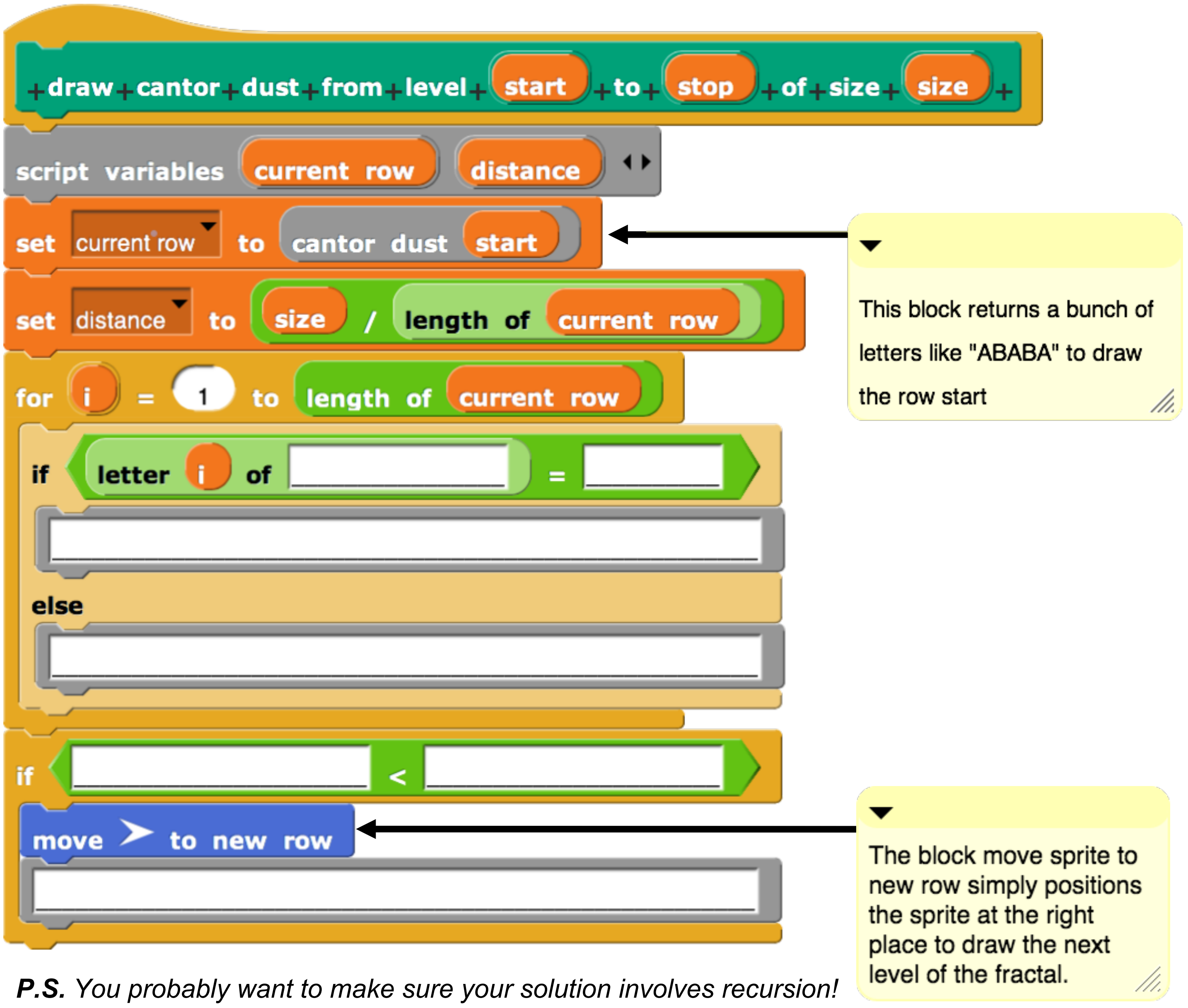
Each letter is a direction for the drawing:

- If the letter is **`'A'`** → draw a line.
- If the letter is **`'B'`** → move (without drawing a line).

There are no other rules or letters you need to worry about.

Your job is to fill in the blanks (on the next page) so that the block will draw the correct image. Below we've provided a collection of Snap*!* blocks to make things easier. You will *not* need to use all the blocks, but all the blocks you need are included. (You may also need to use the variables which we've defined in this block.) All these are blocks you should be familiar with, except for the `draw line of length (n)` block, which simply draws a line of the specified length n.

SID: ______________________________

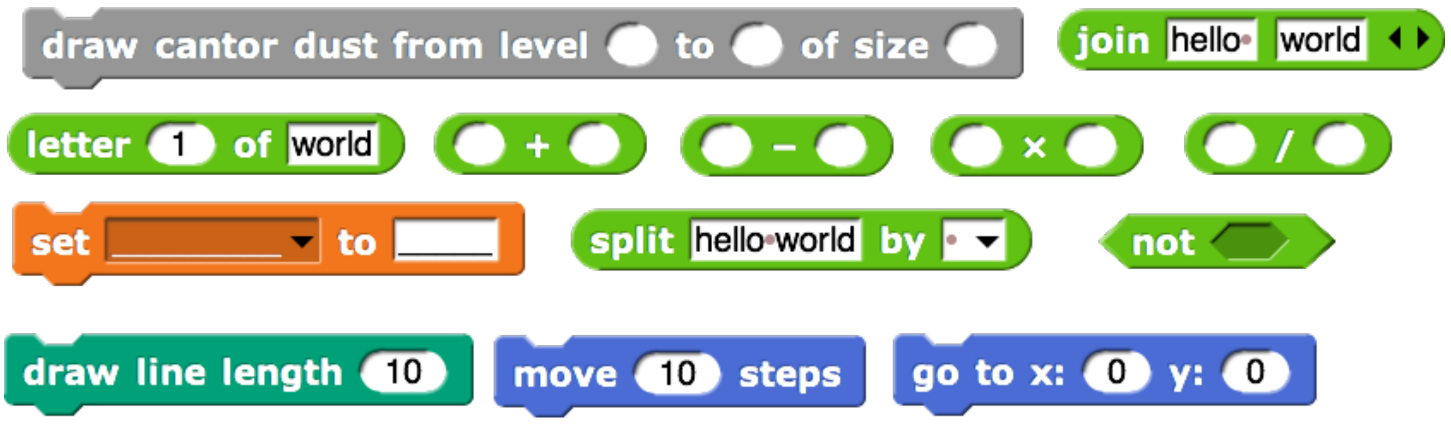


```
draw cantor dust from level (start) to (stop) of size (size)
script variables (current row) (distance)
set [current row] to (cantor dust (start))
set [distance] to ((size) / (length of (current row)))
for (i) = (1) to (length of (current row))
    if <(letter (i) of [ ]) = [ ]>
        [ ]
    else
        [ ]
if <[ ] < [ ]>
    move > to new row
    [ ]
```

This block returns a bunch of letters like "ABABA" to draw the row start

The block move sprite to new row simply positions the sprite at the right place to draw the next level of the fractal.

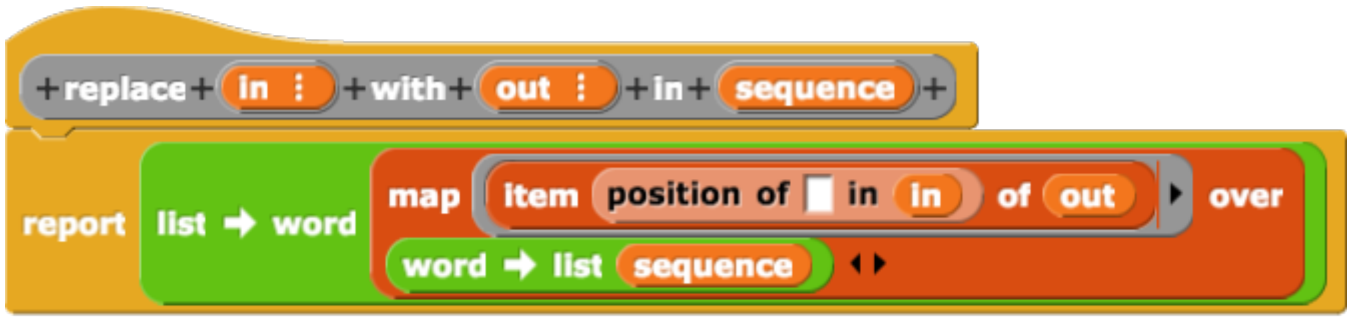
P.S. *You probably want to make sure your solution involves recursion!*



SID: ______________________________

f) This question can be solved independently from the rest of Question 13.

Here is the code that makes up the `replace` block from Part C.



```
+replace+ (in) +with+ (out) +in+ (sequence) +
report (list → word (map (item (position of [ ] in (in)) of (out)) over (word → list (sequence))))
```

What is the runtime of the block `replace (in) with (out) in (sequence)`? (**CLEARLY** circle one.) Read the next paragraph to help you out.

Constant	Logarithmic	Linear	Quadratic	Exponential

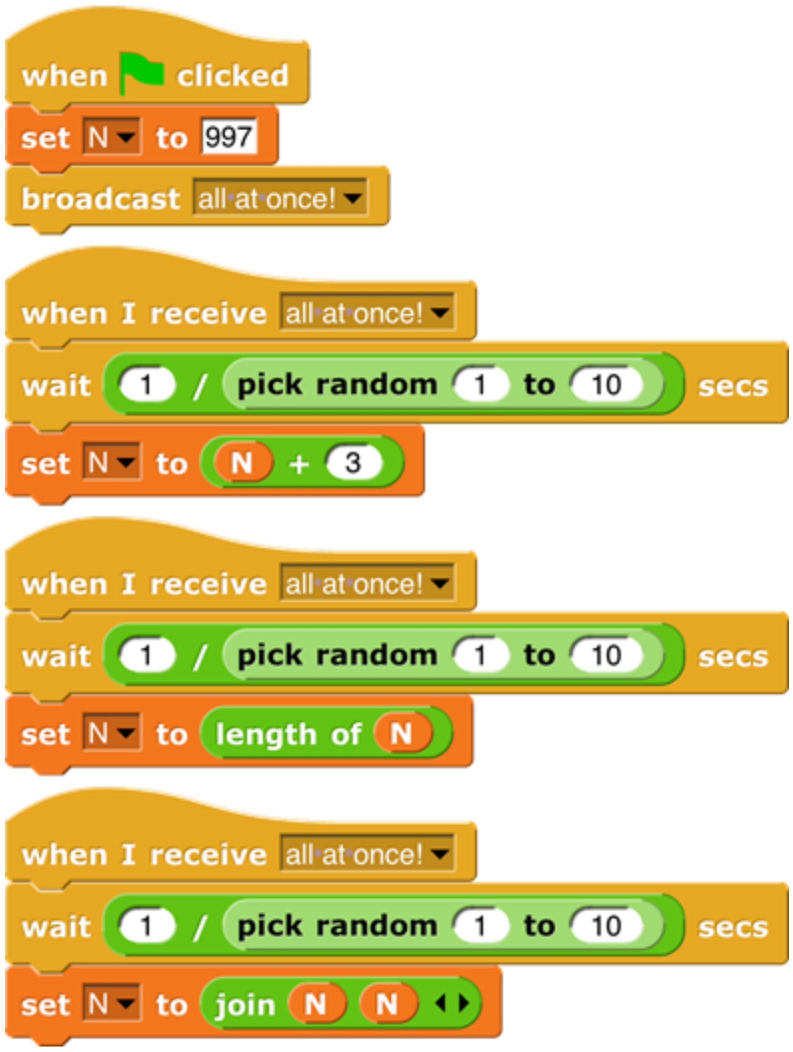
To answer this question, you should know the runtime of these blocks:

• `list → word (list)`	is linear
• `map (function) over (list)`	is linear
• `word → list (word)`	is linear
• `position of (elem) in (lst)`	is linear
• `item (i) of (list)`	is constant

Hint: The runtime of higher-order functions can be a little complex. If you get stuck, try treating the **`map`** block as a **`for`** loop over each letter of the sequence that is using the **`position`** block inside.

SID: ______________________________

Question 14: Visiting a Parallel Universe *(6 points, 12 minutes)*

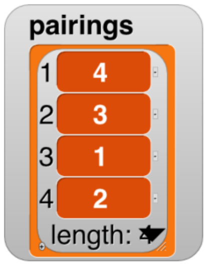


```
when [flag] clicked
set N to 997
broadcast all at once!

when I receive all at once!
wait (1 / pick random 1 to 10) secs
set N to (N + 3)

when I receive all at once!
wait (1 / pick random 1 to 10) secs
set N to length of N

when I receive all at once!
wait (1 / pick random 1 to 10) secs
set N to join N N
```

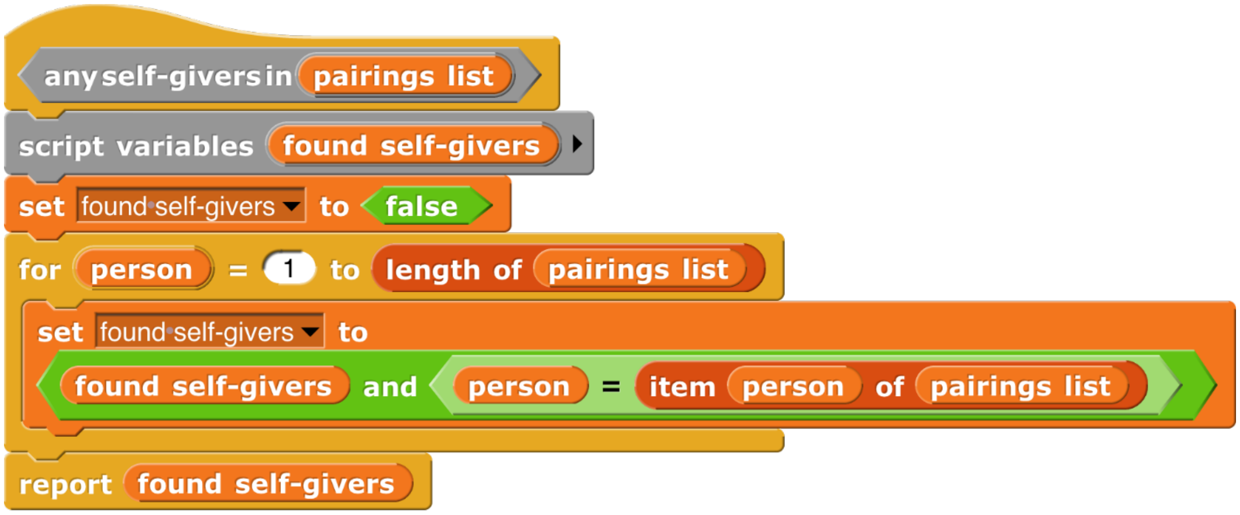
Write all the possible outcomes of the variable **N** when the program runs to completion (that is, when it finishes). Do not write any "intermediate" values of **N** created while the program is running.

SID: ________________________________

Question 15: Holiday Season is Here! *(9 points, 18 minutes)*



You and N of your friends want to set up a 'Secret Santa' situation in which every person is randomly assigned another person in the group, and everyone gives a gift to their recipient. So you all line up, number yourselves from 1 to N, and each person writes their number on a post-it and drops it in a hat. Then everyone closes their eyes, draws a random post-it from the hat, which is the number of the person they'll give to. We encode the pairings as a simple list indicating what person is being given to. For example, the list `(4 3 1 2)` on the right has person 1 giving to person 4 (we'll write 1→4), 2→3, 3→1 and 4→2. (It's basically a permutation of 1-N.) The trouble is, sometimes people inadvertently draw their own numbers! (We call them *self-givers*) We tried to write code to detect whether there are any self-givers in the pairings, but there's a bug.



```
any self-givers in (pairings list)
script variables (found self-givers)
set [found self-givers] to <false>
for (person) = (1) to (length of (pairings list))
    set [found self-givers] to <(found self-givers) and <(person) = (item (person) of (pairings list))>>
report (found self-givers)
```

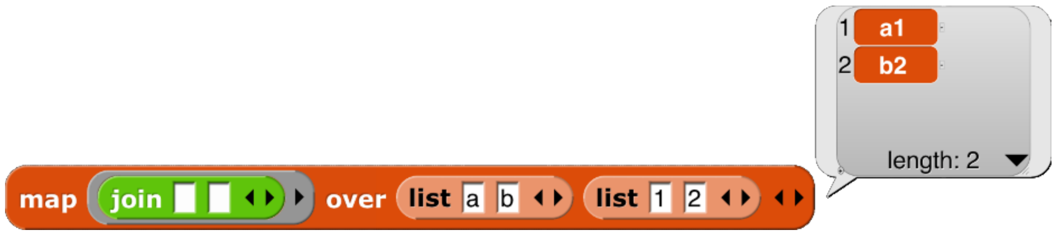
a) Complete the sentence to reveal the bug. The first input should be a list of 4 items.

'Given the following 4-element input list (_________________), `any-self-givers-in(PAIRINGS-LIST)` incorrectly returns (clearly circle one) `true | false` .'

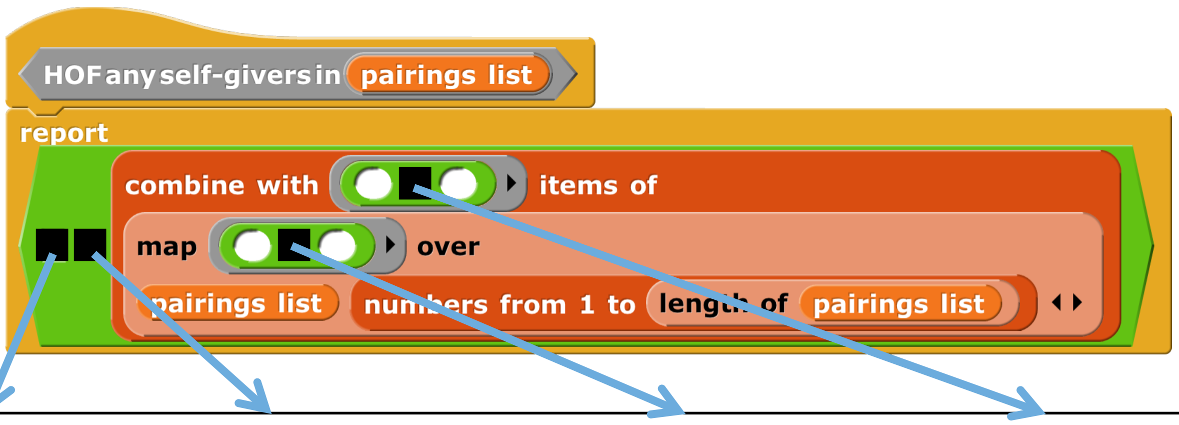
b) Describe the minimal change(s) needed to fix the bug.

SID: ______________________________

c) Fill in the blanks to do it more easily. Let's say we had a block that returned a list of numbers from 1-N in order, called `numbers-from-1-to-(N)`**.** We could combine them with the original pairings using a `map` over 2 lists like this sample.



Fill in the four blanks so that this clever solution works; we give you hints beneath each blank.


HOF any self-givers in pairings list
report
combine with items of
map over
pairings list
numbers from 1 to length of pairings list

_____________	_____________	_____________	_____________
(a number)	(>, < or =)	(-, +, * or /)	(-, +, * or /)

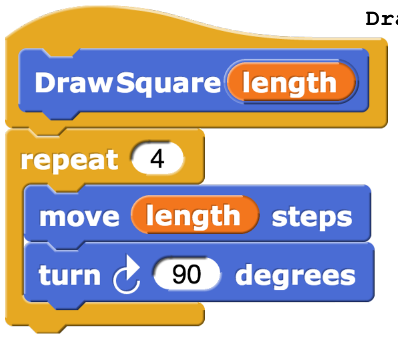
'TWAS THE NIGHT BEFORE CHRISTMAS AT MY FAMILY'S HOUSE.
THERE WERE NO SOUNDS OF STIRRING SAVE THE CLICK OF A MOUSE.
FOR 'TWAS JUST LIKE A CHILDHOOD CHRISTMAS EXCEPT
I'D FORGOTTEN THE HOURS THAT NORMAL FOLKS SLEPT.
WHAT ARE YOU DOING OUT OF BED SO LATE?
LATE? IT'S BARELY 3 AM!

SID: ______________________________

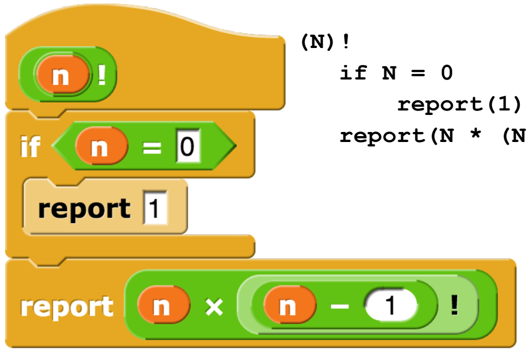
Writing *Snap!* code on paper (supplementary)

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:

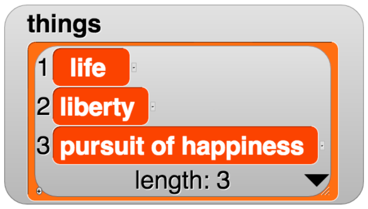


```
Draw-Square(LENGTH)
    repeat(4)
        move(LENGTH)steps
        turn-right(90)degrees
```

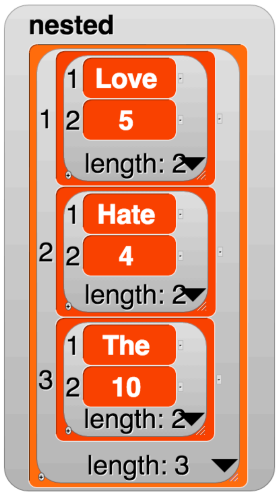


```
(N)!
    if N = 0
        report(1)
    report(N * (N - 1)!)
```

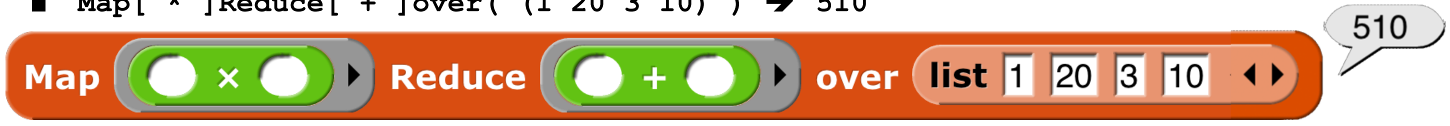
- When you want to write a list of things, write them with an open parenthesis, then the first item, second item, etc (separated by spaces) and when you're done, put a closed parenthesis. If any of your items are a sentence, you have to put quotes around the sentence. So, for example, the following list 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 ring 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 ➔ **`value`**
- So the following would be written as:
  - **`Map[ (_)*(_) ]Reduce[ (_)+(_) ]over( (1 20 3 10) ) ➔ 510`**
- or, in the more simplified (and preferred) format:
  - **`Map[ * ]Reduce[ + ]over( (1 20 3 10) ) ➔ 510`**


Map
×
Reduce
+
over
list 1 20 3 10
510