Midterm Exam CS10 Summer 2017

Name: Solution Student ID: _____ Lab TA: ◯ Jobel ◯ Angela

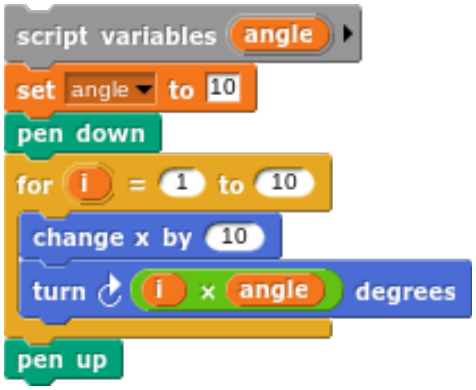
Q1: To Sell or Not to Sell…

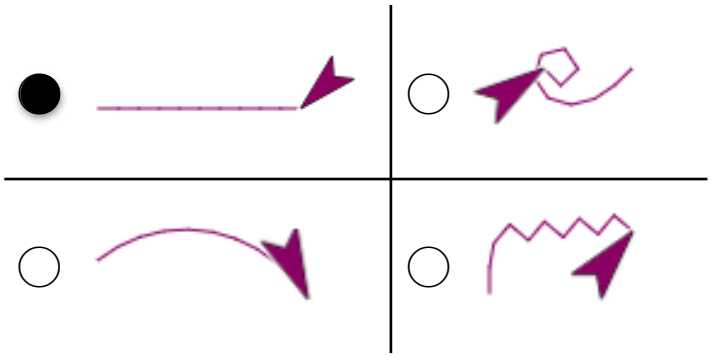
Sally sells seashells by the seashore, but only if the conditions are satisfactory. She only sells shells **if there's sunshine**. She also will only sell shells **if there are no sandcrabs**. Which of the following logical expressions represents when Sally sells seashells by the seashore? Assume sunshine and sandcrabs are boolean variables. (select all that apply)

- ■ sunshine and not sandcrabs
- ☐ sunshine or not sandcrabs
- ☐ not sunshine or sandcrabs
- ☐ not sunshine and sandcrabs
- ■ not not sunshine or sandcrabs

Q2: Drawing Shapes is Fun!

Which drawing will the following script produce? The sprite begins facing up. (pick one)





Q3: Privacy

Which of the following options is/are **false**? (select all that apply)

- ■ You can avoid having an information footprint by not going online.
- ■ If a website says you are anonymous, then your real world identity can remain secret.
- ☐ You can reduce your information footprint by sharing less online.
- ■ HTTPS lets other people listen to your communications over a network

Q4: Abstraction

Which of the following options is **not** an example of abstraction? (pick one)

- ◯ Writing a function that can be called on any input value.
- ◯ Calling an iPod a "music player" instead of an "mp3 player".
- ◯ Representing train routes as straight, perpendicular lines in a train system map.
- ● Writing the recipe for baking a strawberry banana pie using a medium-sized oven.
- ◯ Calling a car's right pedal the "acceleration pedal" instead of the "gas pedal".

Q5: Bits, Nibbles, Bytes

a) Our CS10 class has 53 students. We'd like to give each student an ID number written in binary. What is the least number of bits we need in order to be able to represent 53 unique numbers?

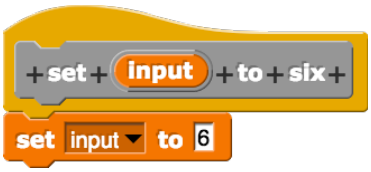
6

b) What is the value of **0b10101** in decimal? 21

Student ID: _____

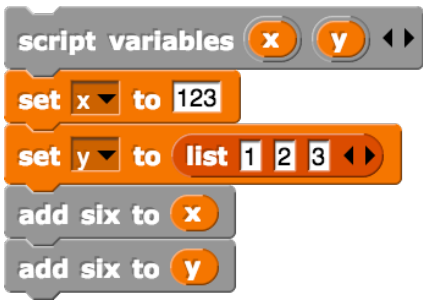
Q6: Mutability

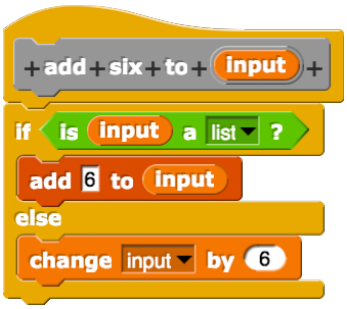
a) What are the values of the script variables x and y after the given script finishes running? (pick one)



- ● x: 123 y: [1, 2, 3]
- ○ x: 6 y: 6
- ○ x: 6 y: [1, 2, 3]
- ○ x: 123 y: 6

b) What are the values of the script variables x and y after the given script finishes running? (pick one)





- ○ x: 123 y: [1, 2, 3]
- ○ x: 129 y: [1, 2, 3, 6]
- ○ x: 129 y: [1, 2, 3]
- ● x: 123 y: [1, 2, 3, 6]

Q7: Cyberpolitics

Why is an attack on critical infrastructure considered one of the most serious cyberattacks? (pick one)

- ○ It could reveal private data
- ○ It could reveal the cyberattack capabilities of a state or government
- ○ It could violate the cyberspace of a country
- ○ It could halt the development of nuclear weapons
- ● It could leave thousands to millions without power, causing massive loss of life and economic damage

Q8: I Coulda Been a Con-Tester

We want to write a block that takes a list and reports the longest chain of repeated values. For example, given the list **[A, B, A, A, A, B, B]**, it should report **3** because there are three A's in a row. It should work for any data types in the given list.
To practice Test Driven Development, you will write unit tests for the block first. You are given an example test case. Come up with four more test cases that test *different* scenarios. Explain what the scenario is that you are testing each time. There are many potential correct answers to this problem.

	Input	***Output***	***Explanation***
Example:	[A, B, A, A, A, B, B]	3	"Longest chain is in the middle of the list"
(a)	**[A, A, A, B, B, B, C]**	**3**	**There are multiple chains that are the longest.**
(b)	**[ ]**	**0**	**The list is empty.**
(c)	**[A, B, C]**	**1**	**All chains have the same length.**
(d)	**[A, A, B, B, C, C, C]**	**3**	**The longest chain is in the end of the list.**

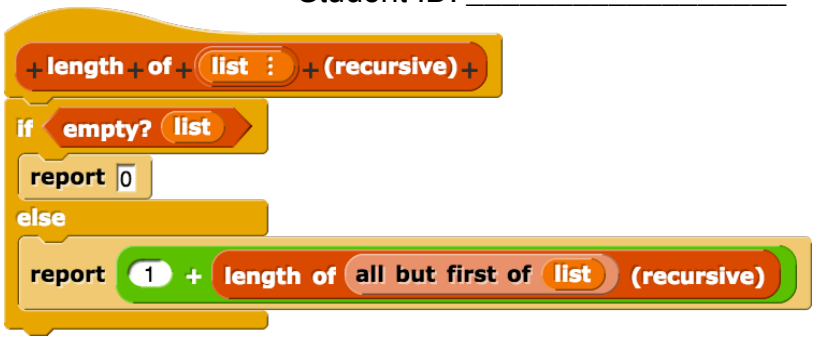
Other examples: whole list same value; different data types; mixed data types; longest at beginning; …

Student ID: _____

Q9: Recursive Order of Growth

What is the number of recursive calls to the **length of *list* (recursive)** block when we call the block as shown below? (pick one)

○ 1 ○ 2 ● 4 ○ 8 ○ 16



Q10: Runtime Analysis

In Homework 2, you wrote algorithms for validating a key, and for encrypting/decrypting a message. Consider the following descriptions of different valid key algorithms. Mark each algorithm's worst case run time *with respect to the length of the input word*. (pick one for each of the four columns)

Valid Key 1: This algorithm reports True if none of the characters in the input word are in a "restricted characters" list. To check, this algorithm starts with the first letter of the input word, and compares it to each item of the "restricted" list. If there is a match, the algorithm reports False. Otherwise, repeat with the second character in the input word. Repeat for all characters.

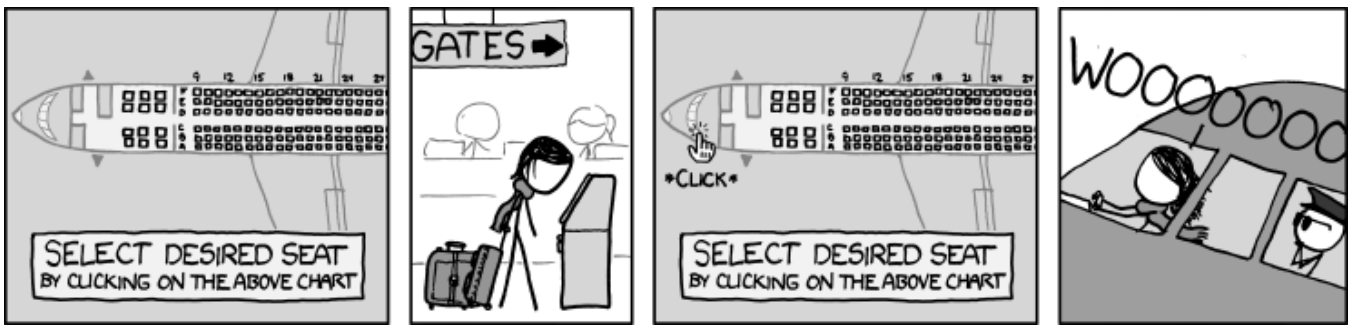
Valid Key 2: This algorithm reports True if all of the characters in the input word are unique. This algorithm compares each character in the input word with every character *after* it. As a result, no comparisons are done twice. If any characters are equal, the algorithm stops and reports False. Otherwise, it reports True at the end.

Valid Key 3: This algorithm reports True if the input word contains the letter "m". For this algorithm, assume we have a helper block that can sort a word alphabetically in constant time. The algorithm searches for the letter "m" as efficiently as possible in the sorted word.

Valid Key 4: This algorithm reports True if there are less than six characters in the input word. It counts each letter in the input word and reports True if the total is less than six, or it stops and reports False if it gets to six letters.

	Valid Key 1	Valid Key 2	Valid Key 3	Valid Key 4
constant	○	○	○	●
logarithmic	○	○	●	○
linear	●	○	○	○
quadratic	○	●	○	○
exponential	○	○	○	○
something else	○	○	○	○

comic relief :)



Q11: Perfect Numbers

Student ID: _____

Introduction

A *perfect number* is a positive integer N that equals the sum of all of its positive divisors (otherwise known as factors: integers that divide it with no remainder), excluding itself. For example, 6 is a perfect number since its positive divisors are [1, 2, 3] and 6 = 1 + 2 + 3. So is 28, since its positive divisors are [1, 2, 4, 7, 14] and 28 = 1 + 2 + 4 + 7 + 14.

Problem

We would like to create a predicate block 

which takes in a number N and outputs True if N is a perfect number and False otherwise. We are given the optional helper block detailed below.

Helper Block	Input(s)	Output
	A positive integer N.	A list containing all positive divisors of N, excluding N.

Using the given helper block and any other Snap! blocks, complete the definition of the perfect number block on the lines below. You may not need to use all of the lines provided. A set of commonly used Snap! blocks is provided to help you, not limit you. There are many potential correct answers to this problem.

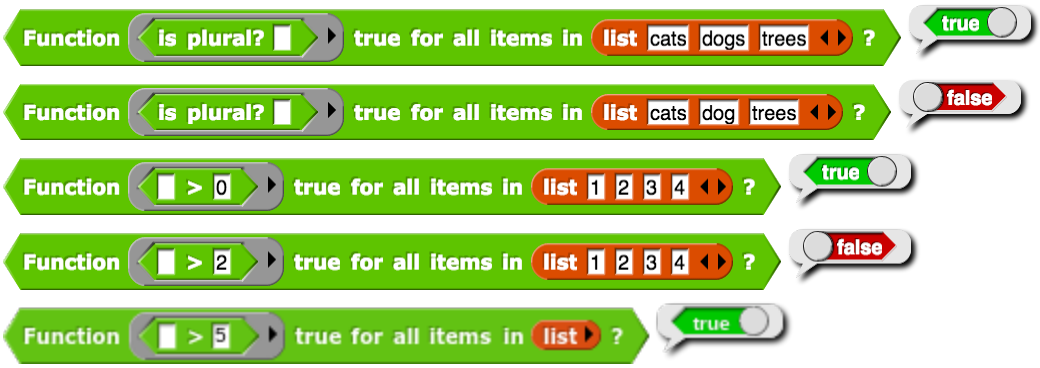
report (n = (combine with (_ + _) items of (all factors of (n))))

Student ID: _____

Q12: Creating a Function of the Higher Order

We would like to create the predicate HOF block 

which takes in a predicate function and a list. The predicate function should take a single argument. Our HOF block should report True if every item in the given list makes the given predicate function report True. Below are some examples. The output of the block is True when the input list is empty, regardless of the predicate.



Let's explore four different ways to build this block.

12a)

The above script uses… (pick one)

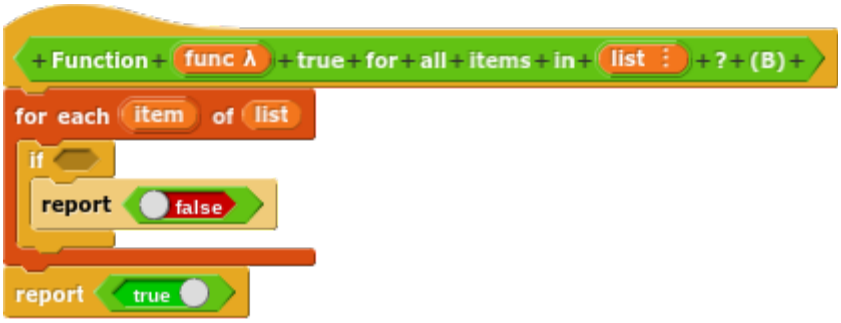
- ◯ Recursion
- ◯ Iteration
- ● Higher Order Function
- ◯ None of the above

The first input to KEEP should be filled with… (pick one)

either

- ◯ call func
- ● call func with inputs
- ◯ run func with inputs
- ◯ map func over list
- ● func

12b)



The above script uses… (pick one)

- ◯ Recursion
- ● Iteration
- ◯ Higher Order Function
- ◯ None of the above

The input to IF should be filled with… (pick one)

- ◯ call func with inputs item
- ◯ map func over item
- ● not call func with inputs item
- ◯ not map func over item
- ◯ func

Student ID: _____

12c)

The script to the right uses… (pick one)

- (●) Recursion
- ( ) Iteration
- ( ) Higher Order Function
- ( ) None of the above

The input to the first IF should be filled with…

- (●) empty? list
- ( ) not empty? list
- ( ) call func with inputs item 1 of list
- ( ) not call func with inputs item 1 of list
- ( ) func

The input to second IF should be filled with…

- ( ) empty? list
- ( ) not empty? list
- (●) call func with inputs item 1 of list
- ( ) not call func with inputs item 1 of list
- ( ) func

12d)

The script above uses… (pick one)

- ( ) Recursion
- ( ) Iteration
- (●) Higher Order Function
- ( ) None of the above

The input to COMBINE should be filled with…

- ( ) join
- ( ) or
- ( ) not
- (●) and
- ( ) true

Q13: Domain and Range

Select the options to fill in the blanks so that the entire expression reports True. The **thing** variable is set as shown below. A reminder of how "input names" works is shown to the right.

Example use of the "input names" functionality from lecture.