

# CS10 Quest- Summer 2019

\_\_\_\_\_  
Your Name (first, last)

\_\_\_\_\_  
ID Card Number

\_\_\_\_\_  
Your 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)

## **Questions 1-2: What's that Smell? It's Potpourri! (2 points each, 4 points total)**

**Question 1:** Which of the following is **not** an example of Abstraction?

- The pedal that makes a car move is called the "accelerator" instead of "gas pedal", because we don't need to know if the car actually uses gas.
- Rather than giving different directions of how to feed each animal, we can just give a single direction of how to feed any type of animal.
- We want our map of the train station to have as many details as possible to ensure we don't get lost.
- We can use a smartphone without understanding the code and hardware that make it work.
- All of the above are examples of Abstraction.

**Question 2:** We are writing a **function** to report the *absolute value* of a number. We test it with the input "-10" and our **function** correctly reports "10" as the *absolute value*. From this, we know that ...

- Our function is correct.
- Our function works correctly for any number input, but we don't know what it will report for text inputs.
- Our function works correctly for any negative number input, but we still don't know if it works for positive numbers.
- Our function works correctly for -10, but we still don't know if it works for other inputs.
- We can't determine any of the above.

## **Question 3: Number Conversions (4 points)**

What is  $12_{10} + 121_3$  in hexadecimal? Write your answer in the box to the right.

**Question 4: Ands and Ors Galore (5 + 2 = 7 points)**

For this problem, consider the block below:



- a. Assuming this expression has no errors, what do we **definitely** know about each of the variables and blocks above? You may select multiple choices per row and column.

	Numbers	Text	Booleans	Lists	Not enough information
Data type of <b>x</b>	<input type="checkbox"/>				
Data type of <b>y</b>	<input type="checkbox"/>				
Domain of bar	<input type="checkbox"/>				
Range of foo	<input type="checkbox"/>				
Range of bar	<input type="checkbox"/>				

- b. Now, assume that the whole expression outputs true. Which of the following do we definitely know?

- must report true
- must report false
- must report true
- must report false
- It is impossible for the expression to report true (do not select any of the previous answers if you select this one)
- None of the above

**Question 5: Lisztomania (2 points each, 8 points total)**

In the box to the right of each script below, write down what the sprite would say after running. You may assume that each script is run independently.

To write down a list, separate each item with commas. For example,  can be written as 1, 2, 3.

a.

```

script variables x y
set x to list 1 2 3
set y to x
delete 1 of x
say x for 2 secs
  
```

b.

```

script variables x y
set x to list 1 2 3
set y to x
delete 1 of x
say y for 2 secs
  
```

c.

```

script variables x y
set x to 1
set y to x
change y by 1
say x for 2 secs
  
```

d.

```

script variables x y
set x to list 1 2 3
set y to x
set y to list a b c
say x for 2 secs
  
```

**Question 6: The Magical Mystery Machine (2 points each, 16 points total)**

For each of the following functions, indicate both what it reports and what the running time is.

--	--	--	--

What does each function report? You may assume A is a positive integer. Choose one option for each row below.

	1	A	2A	A <sup>2</sup>	A <sup>2</sup> + A	A <sup>3</sup>	2 <sup>A</sup>
Mystery1 A	<input type="radio"/>						
Mystery2 A	<input type="radio"/>						
Mystery3 A	<input type="radio"/>						
Mystery4 A	<input type="radio"/>						

What is the running time of each function? Choose one option for each row below.

	Constant	Logarithmic	Linear	Quadratic	Exponential
Mystery1 A	<input type="radio"/>				
Mystery2 A	<input type="radio"/>				
Mystery3 A	<input type="radio"/>				
Mystery4 A	<input type="radio"/>				

**Question 7: An Expansion of Expand (6 points)**

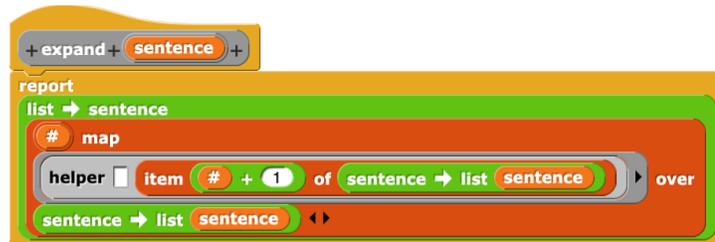
Recall the **expand** block from lab, which takes in a sentence and changes it based on the following rule: if a word is directly after a number, the word is repeated that number of times and the number does not appear in the output. Here are some example calls to **expand**:



We'd like to rewrite the **expand** block, this time using a special version of **map**. This version of **map** has an additional variable, **#**, which refers to the index of the element. Here are some example calls to this special **map**:



Here is our new **expand** block, written with the special **map**:



Write out the code for **helper** in the box below to make **expand** work properly. Feel free to use the following block in your code, which takes in an element and repeats it a certain number of times:



Hint: Since **#** represents the index of the current element, **# + 1** will represent the index of the next element. If the current element is the last element of the list, **# + 1** will be the index of an element that doesn't exist! For this problem, you do not need to worry about this and can trust that the given code does not error.

Implement the function **helper** in the box below so that **expand** works as explained.

**helper** (input1) (input2):

**Question 8: Pair-Mutations (6 + 6 = 12 points)**

We are trying to write a command block called **swap pairs** that will have two lists as input. Both lists will contain numbers, and both lists **must** have the same length. Each **pair** is the two items at the same index from the two lists. (eg. item 1 of list **A** and item 1 of list **B** form a **pair**, item 2 of list **A** and item 2 of list **B** form a different **pair**, etc.) For every pair in our lists, we want this block to put the smaller value into list **A** and the larger value into list **B**. If both items in the pair have the same value, then the values in that pair remain the same.

For example, after running the following code:



list **A** should have the values: 

list **B** should have the values: 

(No values were changed because for each **pair**, the smaller value was already in list **A**)

After running this code:



list **A** should have the values: 

list **B** should have the values: 

(The values are switched in every **pair** because  $4 > 1$ ,  $5 > 2$ ,  $6 > 3$ .)

After running this code:



list **A** should have the values: 

list **B** should have the values: 

(For the **pair** at the first index,  $1 < 2$  so the values stay the same. For the **pair** at the second index,  $5 = 5$  so the values stay 5. For the **pair** at the third index,  $10 > 8$  so the values are switched.)

- a. We started writing the block, but it doesn't always work correctly! To help us debug our code, indicate what the resulting values will be for list **A** and list **B** using our **incorrect** block. Here is our block:

```

+ swap + pairs + (buggy) + list1 + list2 +
for i = 1 to length of list1
if item i of list1 > item i of list2
  replace item i of list1 with item i of list2
  replace item i of list2 with item i of list1

```

What will be the values of **A** and **B** after running the following 3 scripts? Write your answers directly into the empty slots of the *list* blocks.

	A	B
<pre> set A to list 1 2 3 set B to list 4 5 6 swap pairs (buggy) A B </pre>	<pre>list [ ] [ ] [ ]</pre>	<pre>list [ ] [ ] [ ]</pre>
<pre> set A to list 4 5 6 set B to list 1 2 3 swap pairs (buggy) A B </pre>	<pre>list [ ] [ ] [ ]</pre>	<pre>list [ ] [ ] [ ]</pre>
<pre> set A to list 1 5 10 set B to list 2 5 8 swap pairs (buggy) A B </pre>	<pre>list [ ] [ ] [ ]</pre>	<pre>list [ ] [ ] [ ]</pre>

- b. Now that we know how our buggy solution behaves, it's time for you to help us create a working block! On the lines below, write out code to implement a working version of `swap pairs`. Just like in the original code, your block must mutate the original lists: it may not product new lists. You may only use the lines given, though you may not need to use all of them. Please write clearly!

**`swap pairs (list1) (list2):`**

---

---

---

---

---

---

---

---

## Writing Snap! Code on Paper (Supplementary)

In CS10 exams, you may be asked to write Snap! code on paper. Follow the guidelines below to write out your code. If you have any questions, feel free to ask us!

1. To write out nested blocks, use parentheses:

 would be written out as  $(2 \times 3) + 4$

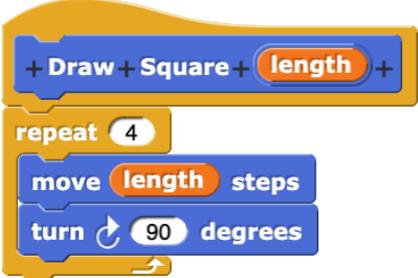
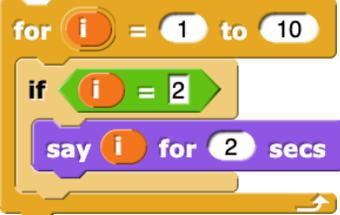
2. You may represent variables by putting parentheses around their names:

 would be written out as  $(a) + (b)$

3. Lists are written out with square brackets around them and commas between items:

 would be written out as  $[1, 2, 3]$

4. Use indentation to indicate what code belongs to a function or control structure. Below are some examples of what this would look like:

 <pre> Draw Square (length)   repeat 4     move (length) steps     turn 90 degrees clockwise           </pre>	 <pre> for (i) = 1 to 10   if (i) = 2     say (i) for 2 secs           </pre>
---	--

3. If you want to leave an input slot blank, replace it with  $()$

 would be written as  $4 \times ()$

Below are some examples of Snap! code written out:

	<pre>map (() x ()) over [1, 2]</pre>
	<pre>script variables (a) set (a) to 1</pre>

## Useful Blocks

A bunch of Snap blocks are shown below as a reference. For coding problems on this exam, unless the problem says otherwise, you may use any Snap! block, not just the ones below (we've omitted lots of them, like x, =, split, etc.). The values input in these blocks are default inputs; you may change them.

