Discussion 14: Final Review

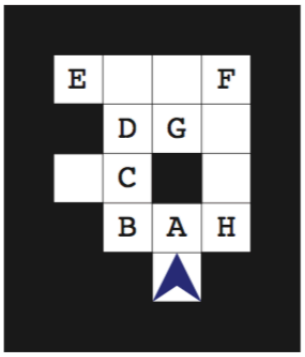
Binary

1. YouTube now uses 64 instead of 32 bits to count views. How many more is that? 2^{32} times more

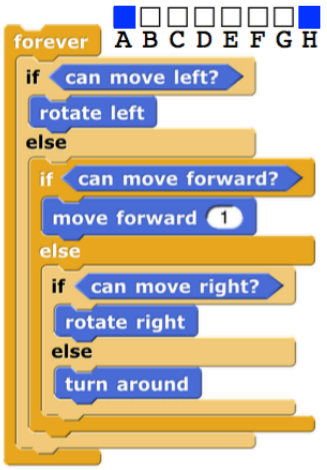
Drawing/Movement in Snap

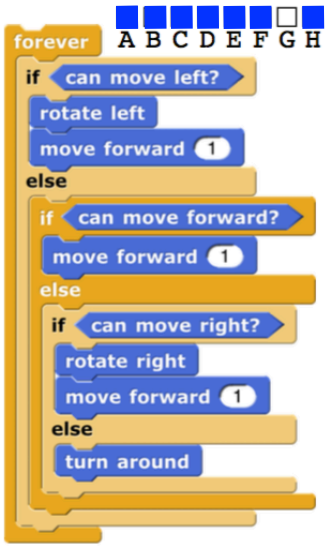
Question 1: Mr. Robot 2

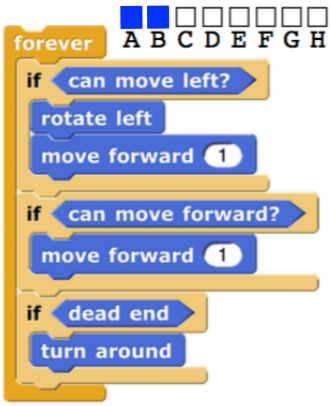
We tried to rewrite our midterm maze script to visit all the letters A-H in the maze. Here are our four attempts, let us know the letters they each visit.

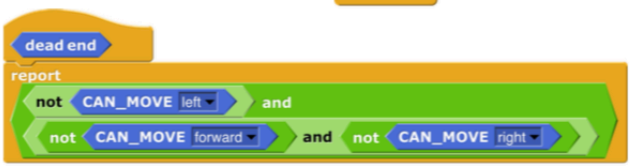


move forward	rotate left rotate right turn around	can move left? can move forward? can move right? dead end
The robot moves *INPUT squares* forward in the direction it's facing.	The robot turns, in-place. {left = counterclockwise, right = clockwise, around = u-turn}	Reports `true` if the robot has a free square to its {left, front, right}; otherwise reports false. The last one reports `true` if *can't* move left, forward *and* right.



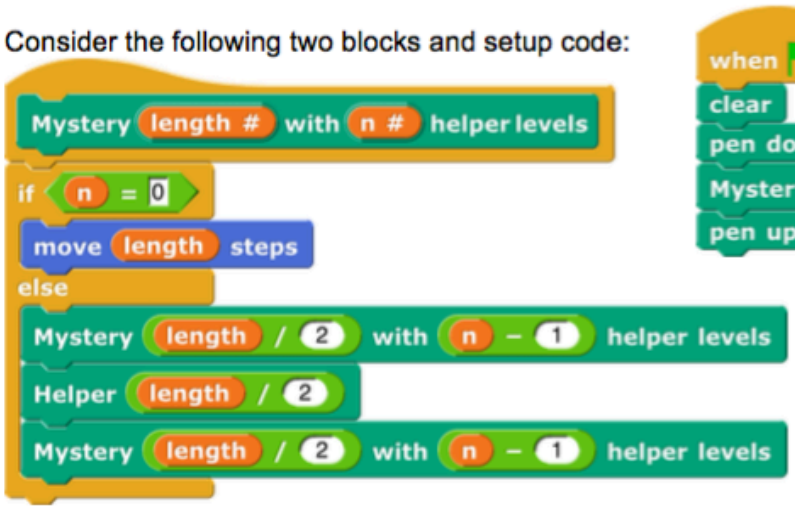


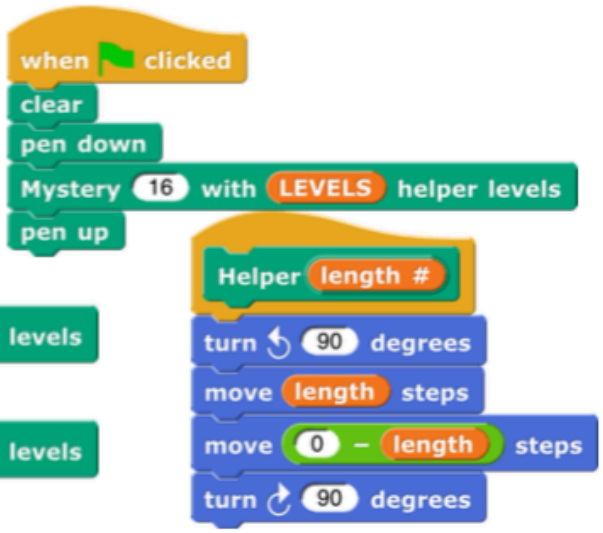




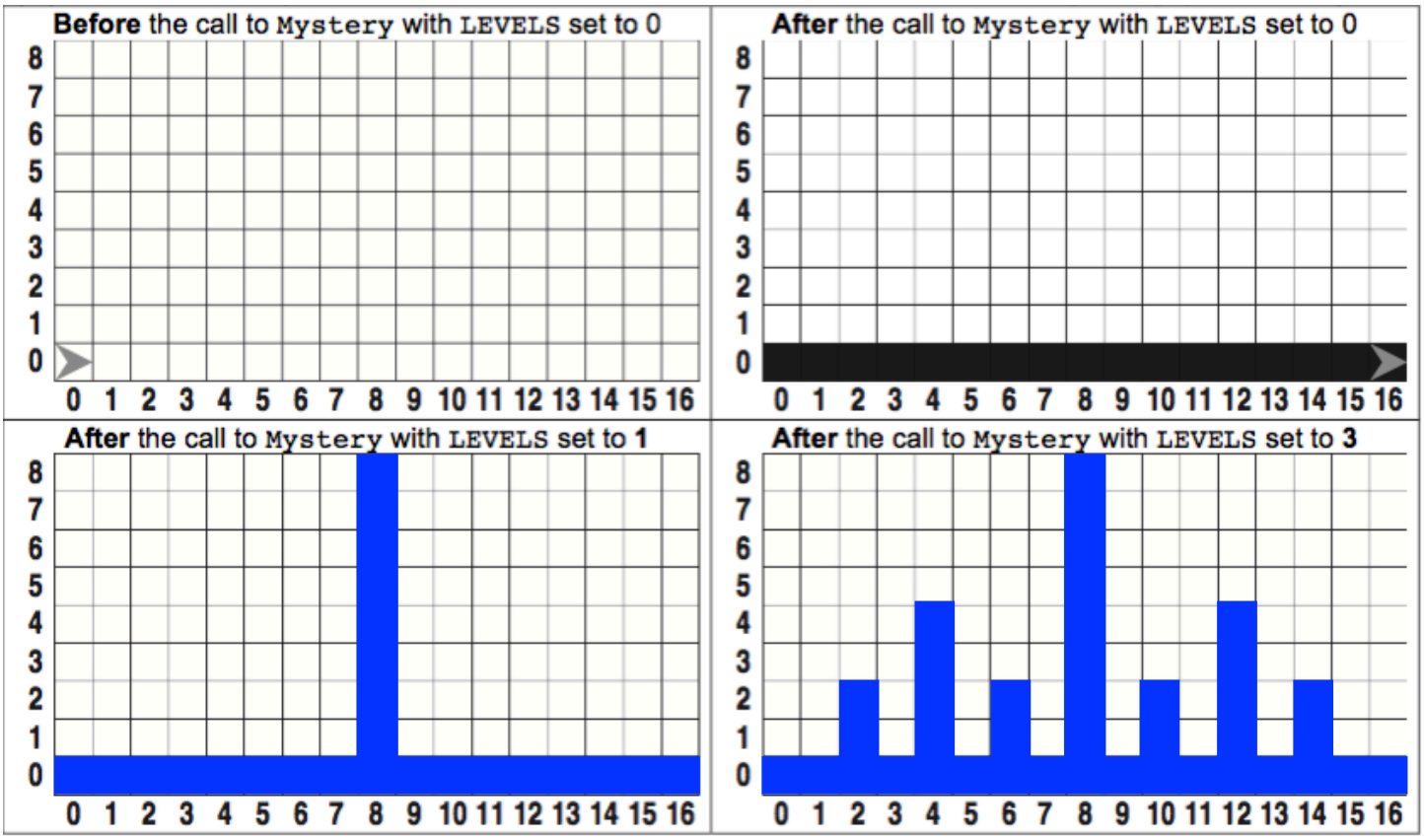
Question 2: Magical Mystery Tour

Consider the following two blocks and setup code:





a. Now, given that the sprite starts out in the bottom left corner facing right, and that the pen is in the middle of the sprite, shade in the pixels that will be colored after calls to Mystery with levels set to 1 and levels set to 3. You may use the top left grid for scratch work. Levels = 0 has been given to you.



b. Runtime

We're told that it actually costs a *dollar* to fill in all the pixels drawn by `Helper`. Which expression best captures the cost (in dollars) for this call? (select ONE)

L	½*L	N	½*N	L*N	½*L*N	L^N	½*L^N	N^L	½*N^L	None of these
○	○	○	○	○	●	○	○	○	○	○

Recursion

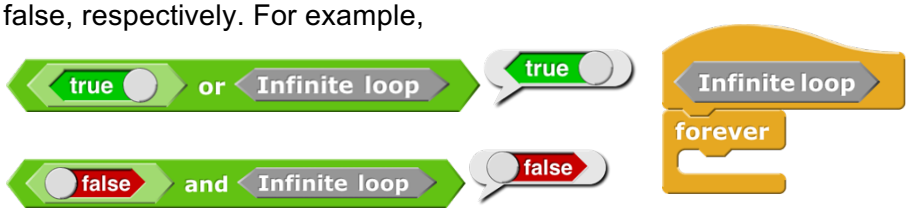
Question 1: Ready, Set, Go!

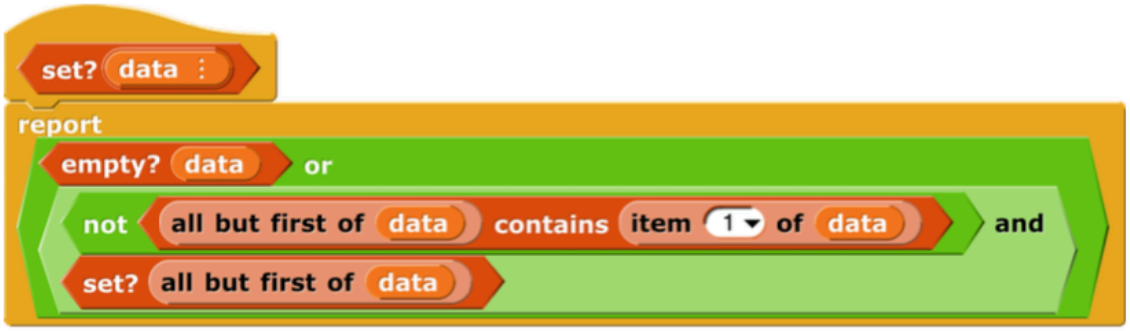
In this problem, we have created three different blocks to see if a given list is a set, that is, it has no duplicates. For each of the blocks below, select one of the following answer choices:

Example calls to set?

A = it works fine.
B = It will cause an error or run forever.
C = It always returns *true*.
D = It always returns *false*.
E = If it's the empty list, *true*, otherwise it always returns *false*
F = If it's the empty list, *false*, otherwise it always returns *true*
G = If it's the empty list, *true*, otherwise it only returns whether the *first* element is in the list multiple times
H = If it's the empty list, *true*, otherwise it only returns whether the *last* element is in the list multiple times

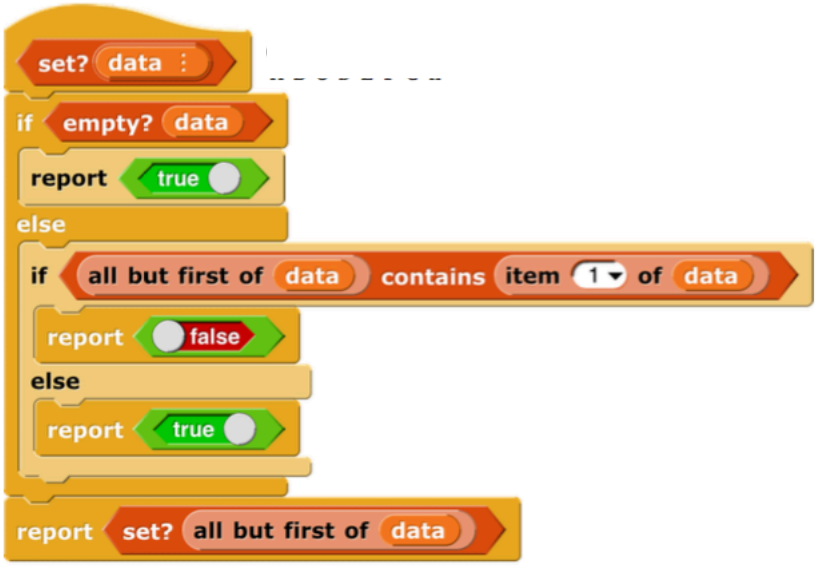
a. For this subpart, note that the *or* and *and* blocks don't even look at their right input if the left one is true or false, respectively. For example,





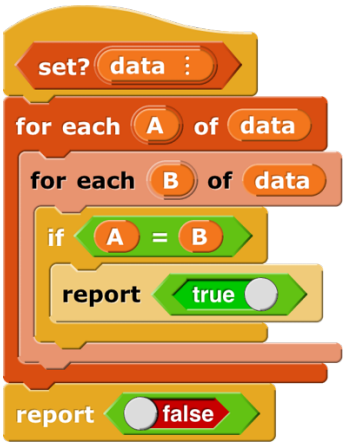
- (●) A
- ( ) B
- ( ) C
- ( ) D
- ( ) F
- ( ) G
- ( ) H

b.



- ( ) A
- ( ) B
- ( ) C
- ( ) D
- ( ) F
- (●) G
- ( ) H

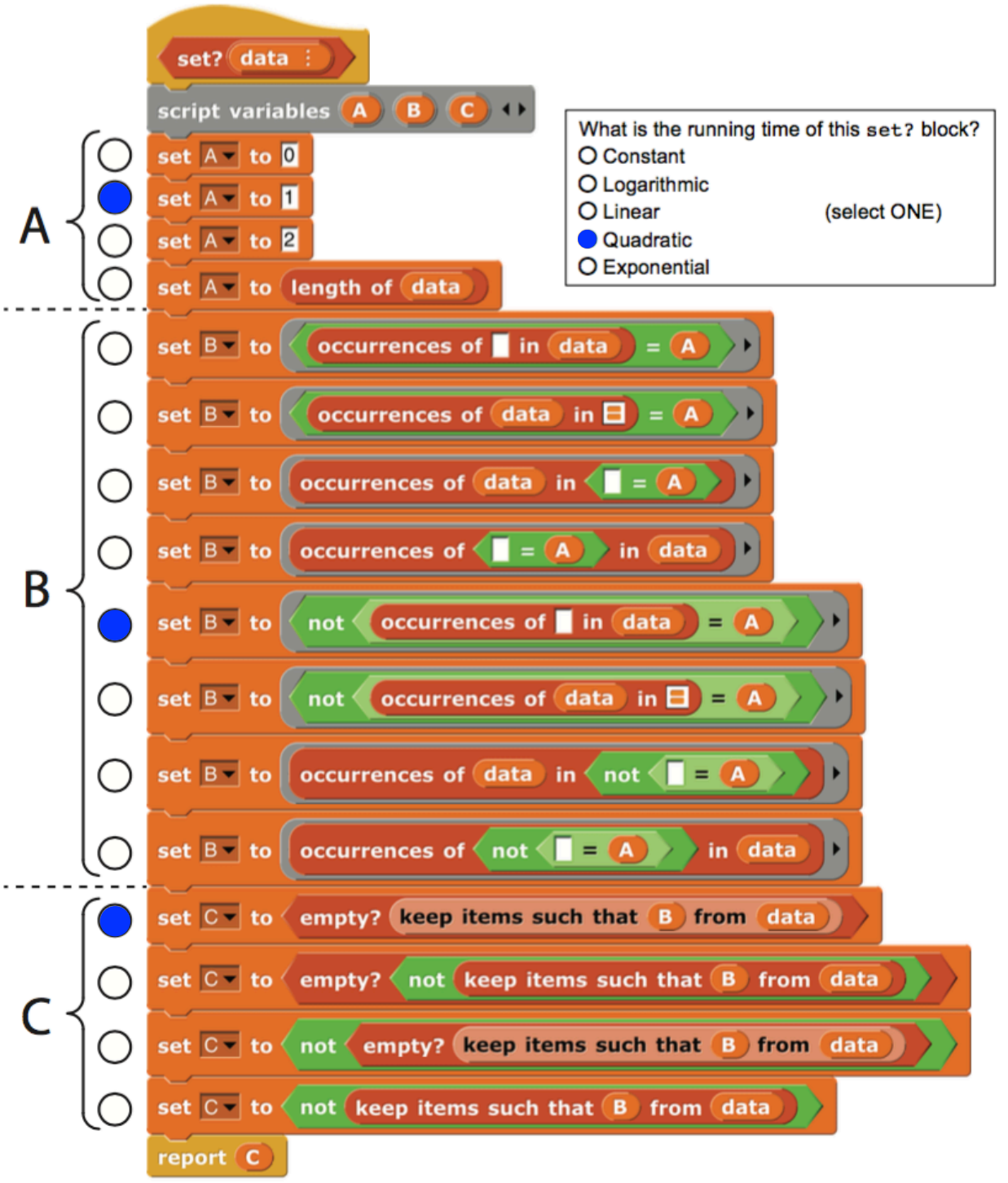
c.



- ( ) A
- ( ) B
- ( ) C
- ( ) D
- (●) F
- ( ) G
- ( ) H

Question 2: Constructing the *set* block

How could we construct the *set* block using the following *occurrences of* block? Note that you may only choose one option from each section A-C.



Python

Question 1: Syntax
Write the output of the following lines of code.

```
>>> ['cal', 'berkeley', 'stanford'][1][2]
'r'
>>> [x*10 for x in range(3) if x != 1]
[0, 20]
```

Question 2: Reversing a Dictionary
We want to write a dictionary reverser that takes in a dictionary and returns a new dictionary with the original values as the new keys and the original keys as a list of values.

```
>>> dictionary_reverser({1:3, 2:3, 8:9})
{3: [1, 2], 9: [8]}
```

Write this function by filling in the blanks in the skeleton code below.

```
def dictionary_reverser(dict):
    r = {}
    for k in dict:
        if dict[k] in r:
            r[dict[k]].append(k)
        else:
            r[dict[k]] = [k]
    return r
```