Discussion 7

## First an example: Factorial

- "the factorial of a non-negative number n, denoted by n!, is the product of all positive integers less than or equal to n"
  - $\circ$  "for example, 5! = 5 x 4 x 3 x 2 x 1 = 120"
- we can also write this as

$$5! = 5 \times 4! = 5 \times 4 \times 3! = 5 \times 4 \times 3 \times 2! = 5 \times 4 \times 3 \times 2 \times 1! = 5 \times 4 \times 3 \times 1! = 5 \times 4 \times 1! = 5 \times 4 \times 1! = 5 \times$$

## **Factorial in Snap**

using factorial (5) as an example

```
+factorial + num +

If num < 2

report 1

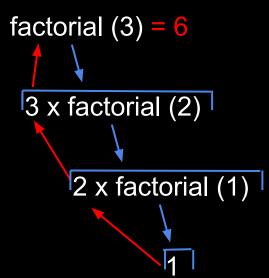
else

report num × factorial num - 1
```

- in the previous slide, 5! could be represented as 5 x 4! = 5 x 4 x 3! and etc.
- the same logic applies for this Snap version
- factorial (5) = 5 x factorial (4) = 5 x 4 x factorial (3), and etc.
- the recursion finally ends at factorial (1), when our block just reports 1, not calling itself again

### Factorial in Snap

 think of this recursive process like a ladder (you go down the ladder until you hit the base case, then you go back up to evaluate and compute the values)



### What does recursion mean to you?

Based on what you've learned in lab, lecture, plus your personal thoughts and/or analogies, how would you define recursion?

Base Case(s):

Recursive Case(s):

Base Case(s):
-Simplest form of the problem

Recursive Case(s):

Base Case(s):

-Simplest form of the problem

Recursive Case(s):

-Divide problem into smaller instances

Base Case(s):

-Simplest form of the problem

Recursive Case(s):

- -Divide problem into smaller instances
- -Invoke function (recursively)

Base Case(s):

-Simplest form of the problem

Recursive Case(s):

- -Divide problem into smaller instances
- -Invoke function (recursively)
- -Work towards base case

So what is the fibonacci sequence?

So what is the fibonacci sequence?

- they are the numbers 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
- "in mathematical terms, the sequence  $F_n$  of Fibonacci numbers is defined by the recurrence relation

 $F_n = F_{n-1} + F_{n-2}$  with seed values  $F_0 = 0$ ,  $F_1 = 1$ "

 $F_n = F_{n-1} + F_{n-2}$  with seed values  $F_0 = 0$ ,  $F_1 = 1$  "

we can turn this recurrence relation into a recursive function

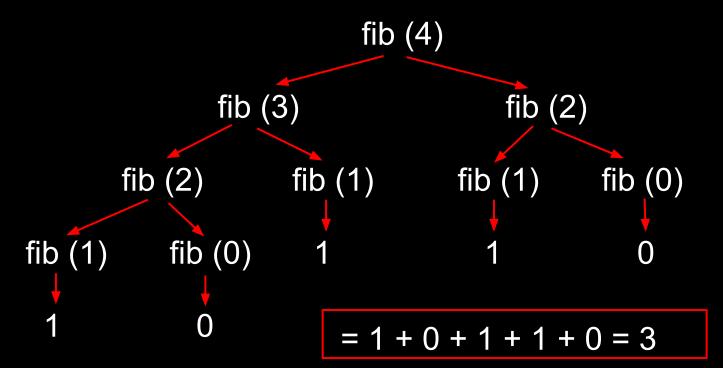
in Snap

```
+ fibonacci + num
report 0
report 1
                                      fibonacci
        fibonacci
```

- the fibonacci block actually has two base cases
- there are also two recursive calls of the fibonacci block in the recursive case

```
+ fibonacci + num
report 0
report 1
                                       fibonacci
        fibonacci
```

\* fib short for fibonacci



## **Practice Problems**

Note: Most of these problems will require you to first import the "List Utilities" and "Words, sentences" libraries. Also remember to use recursion in your solutions.

## Say My Name, Say My Name

Write a block that says all numbers between an input number and 0

```
Say the number from 20 to zero
```

## Say My Name, Say My Name

```
+ Say + the + number + from + num + to + zero +
   num | < 1
say 0 for 1 secs
say num for 1 secs
Say the number from
                                   to zero
```

#### I Got 99 letters but a "B" ain't one

Given a word as input, find the number of characters in the word



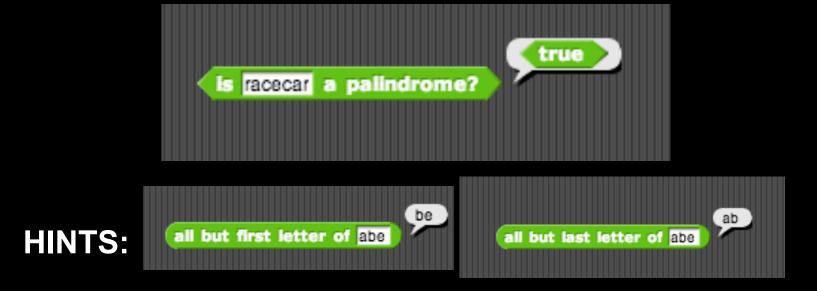
HINT:

all but first letter of abe

#### I Got 99 letters but a "B" ain't one

#### O.N.I.F.C.F.I.N.O.

A palindrome is a word that is spelled the same way forwards and backwards (example: racecar). Given a word as input, report whether or not the word is a palindrome.



#### O.N.I.F.C.F.I.N.O.

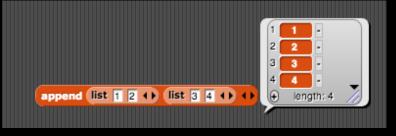
```
+ is + word + a + palindrome? +
    length of word = 1 or length of <math>word = 0
report true
   letter 1 of word = letter length of word of word
 report
  is all but first letter of all but last letter of word
                                                     a palindrome?
else
report < false
```

## **Smack My List Up**

Given a list as input, report the reverse of the list



HINTS:





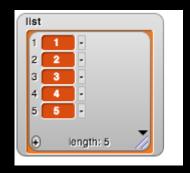
## **Smack My List Up**

```
+ reverse + of + the + list + ( list )+
    length of list
report list
else
report
  append
                                               list Item 1 → of list → ↔
  reverse of the list all but first of list
```

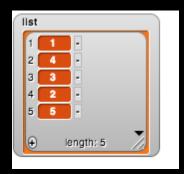
## Smack (Part of) My List Up

Given a list, a minimum number and maximum number as input, reverse the order of the list, but only for items within the range of minimum to maximum.





**Original List** 



List After Block Call

## Smack (Part of) My List Up

```
+ reverse + order + of + (list) + from + (min) + to + (max) +
script variables temp1 temp2 temp2
   min > (max
stop this block ▼
 set temp1 to item (min) of list
 set temp2 to item (max) of list
 replace item (max) of (list) with (temp1
 replace item (min) of (list) with (temp2)
 reverse order of (list) from (min) + (1) to (max) -
```

## Report Length or Die Trying

Write the



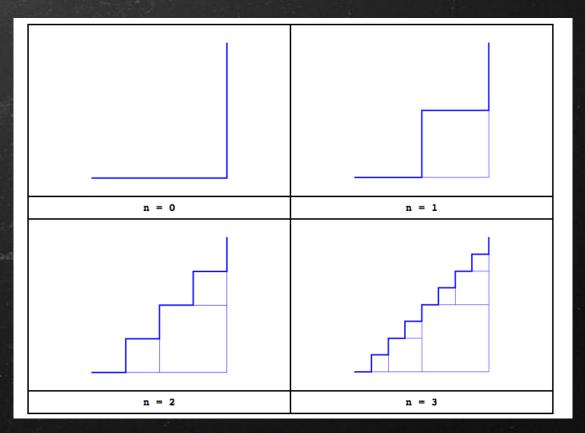
block without using the built-in

Snap! block.

## Report Length or Die Trying

```
+ length + of + list + list : +
   empty?
report 0
else
         1 + length of list all but first of list
report
```

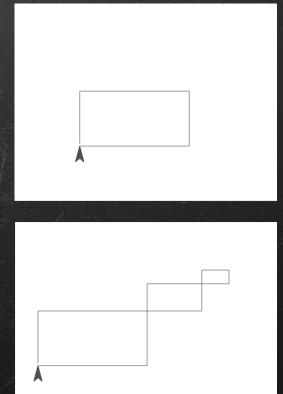
## Ladder Fractal

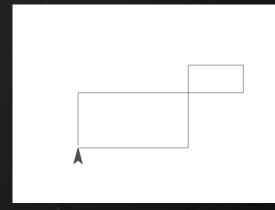


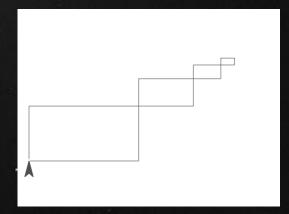
## Ladder Fractal Solution

```
+ ladder + fractal + with + n + levels + of + size + s +
turn 5 90 degrees
turn 5 -90 degrees
ladder fractal with
ladder fractal with n - 1
```

# Rectangle Fractal







## Rectangle Fractal Solution

```
+ Rectangle + fractal + level + levels + and + size + size +
repeat 2
 move size / 2 steps
 turn & 90 degrees
 move size steps
 turn 👌 90 degrees
     size / 2 steps
turn 👌 90 degrees
turn 5 90 degrees
                    levels - 1 and size size / 2
Rectangle fractal level
turn (180) degrees
      size / 2 steps
turn 2 90 degrees
turn 👌 90 degrees
```