Code in Place 2023

Stanford CS106A

Section - Week 2

Deeper Dive Into Decomposition

David Tsai, Code in Place

Today's Agenda

1. Check-In

How are we all doing?

2. Concepts Review

Decomposition,
Pre/Postconditions

3. Practice Problem

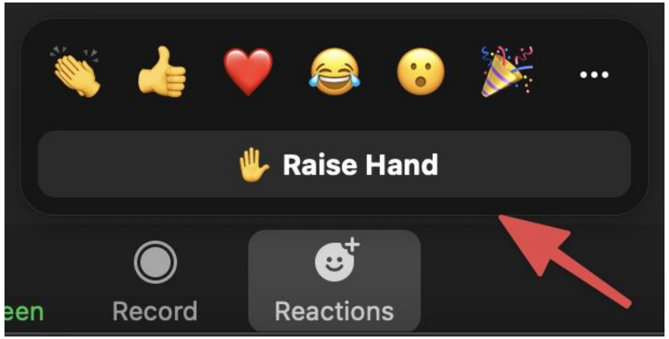
"Spread Beeper"

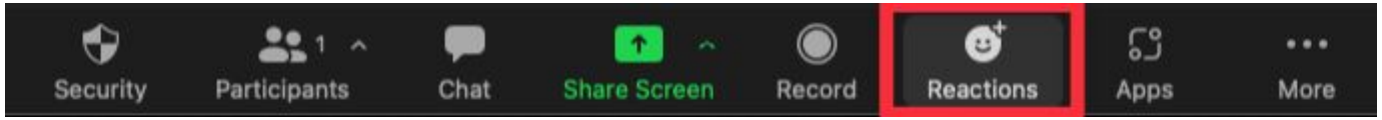
4. Bonus Problems

If we have time

Zoom Reactions

- Thumbs Up: If you understand.
- Raise Hand: If you have a question (or just speak in the mic)





Before We Start

How are you all doing?
Hopefully your first week of CIP went well!

- What’s one thing you have enjoyed or found fun about CIP so far?
- What has been your favorite problem to work on so far?
- Is there anything blocking you for completing Assignment 1?
- Any major questions?

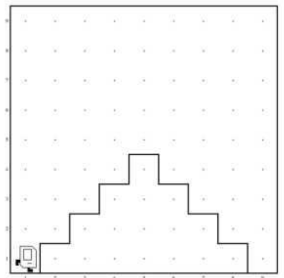
Concepts Review

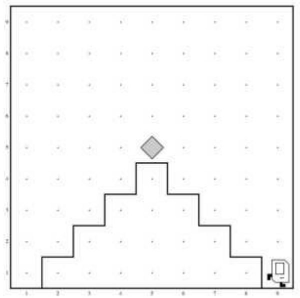
Decomposition

Break down a problem into more manageable sub-problems.

A good function should:

- Do one "conceptual thing"
- Know what the function does, by looking at its name
- Less than 10 lines, and less than 3 levels of indentation
- Reusable and easy to modify
- Well commented (brief description, precondition, postcondition)





```
def main():
        climb_mountain()
        put_beeper()
        descend_mountain()
```

Pre and Postconditions

The **precondition** and **postcondition** are a guarantee to anyone who uses the function that the conditions stated will hold true.

Precondition:

- A condition that must always be true just prior to the execution of a function.

Postcondition

- A condition that must always be true after the execution of a function.

Pre and Postconditions

Questions to ask before a function runs and after it ends:

"Before" a function runs:

- Where is Karel now?
- What direction is Karel facing?
- What is clear? What is blocked?
- Are there beepers present?

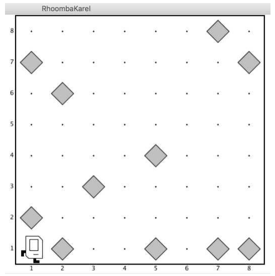
"After" a function runs:

- Is Karel in the same position as it was before?
- Is Karel facing the same direction?
- Is its front/left/right clear or blocked?
- Is it sitting on beeper(s)?
- Will the function *always* finish this way?

Consistency is important:

- Try to define a situation that will be consistent for every iteration through the function
- Look out for edge cases
  - "fencepost" ("off-by-one") errors
  - Typically come up at the <u>beginning</u> or at the <u>end</u> of a run

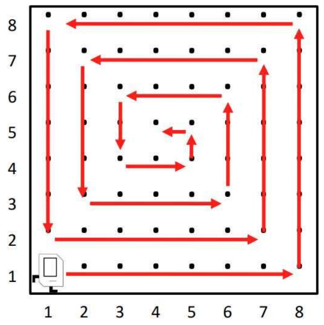
Pick a Strategy That Has Simple Pre/Postconditions



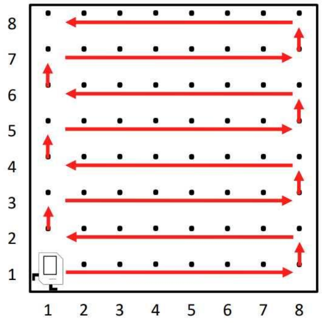
Roomba Karel:
Pick up all the beepers in the world.

Which possible algorithm would you choose?

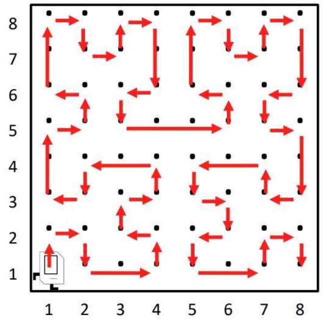
Algorithm 1



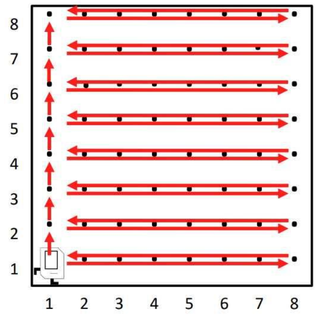
Algorithm 2



Algorithm 3



Algorithm 4

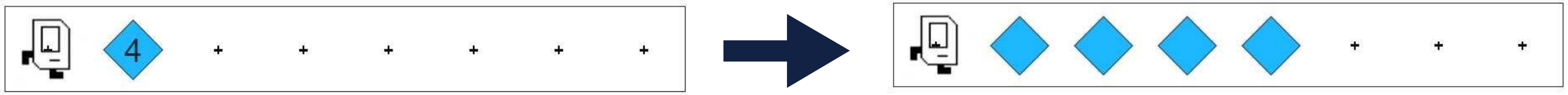


Section Exercise: “Spread Beeper”

Let’s do something a bit more algorithmically interesting and challenging

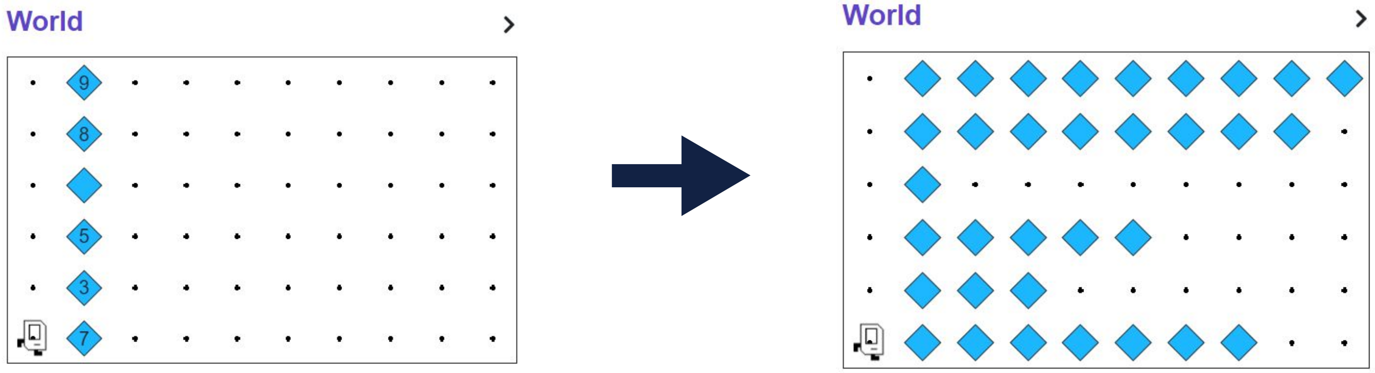
- This problem is intended to be harder than last week’s section problem!

- Outline problem logic
- Apply control flow to execute outline (while loops + conditionals)
- Multiple decompositions
- Think about the postcondition of loops, and if you are ready to repeat
    - Look for possible “fencepost errors” (aka “off-by-one”), such as a special case for the final step



Bonus Challenge: Multiple Rows!

What if you want to extend your code from <u>one row</u> to a world with an <u>unknown number of rows</u>?



Intro to Console Programming

Welcome to real-life Python world!
No longer restricted to Karel's world with only 4 commands!

Basic commands for Bonus Exercise:

`print()`

- Prints text or value to console
- Example:
  - `print("Hello, world!")`
  - `print(42)`

`input()`

- Requests the user to type in an input
- Example:
  - `user_height = input("Please enter your height: ")`
  - `print(user_height)`

Bonus Exercise: Intro to Console

Get a string (a piece of text) from the user, and then print the string 10 times to the Python console.

Input

```
Line: coding rocks!
```

$\longrightarrow$

Output

```
coding rocks!
coding rocks!
coding rocks!
coding rocks!
coding rocks!
coding rocks!
coding rocks!
coding rocks!
coding rocks!
coding rocks!
```