

# Code in Place 2025

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 Problem

If we have time

# Introductions

Hi, I'm David!

- Head TA, here at my 4th Code in Place.
  - Started as a CIP student! 2x volunteer Section Leader.
- CS @ Massachusetts Institute of Technology (MIT)
- Produced Manager and former Software Engineer
- Love photography, video games, movies
- Guilty pleasure: Reality competition shows like Survivor
- Cool superpower I'd like to have:
  - Felix Felicis ("Liquid Luck" potion) from Harry Potter

## 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 from completing Week 1 assignments?
- Any major questions?

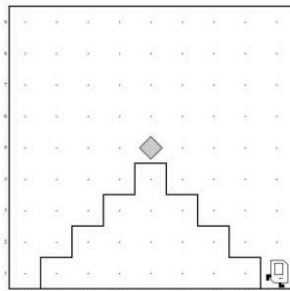
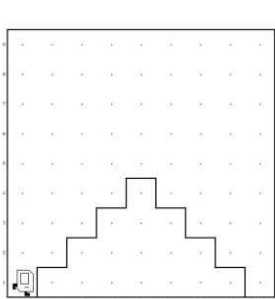
# Concepts Review

# 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?

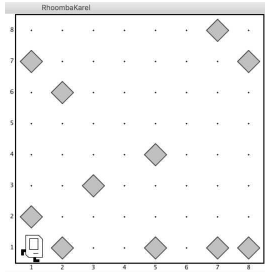


**Consistency** is important:

- 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 beginning or at the end of a run



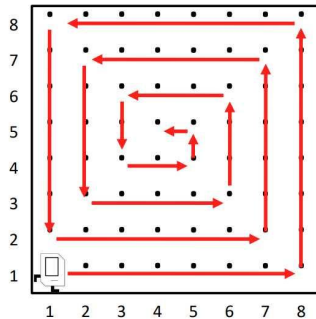
# Pick a Strategy That Has Simple Pre/Postconditions



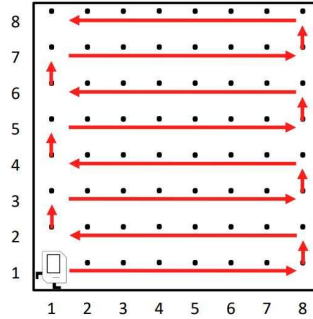
Roomba Karel:  
Pick up all the beepers in the world.

Which possible algorithm would you choose?

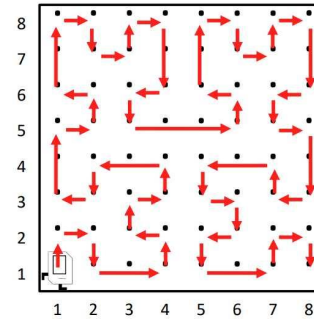
Algorithm 1



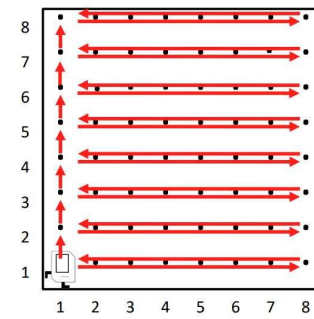
Algorithm 2



Algorithm 3



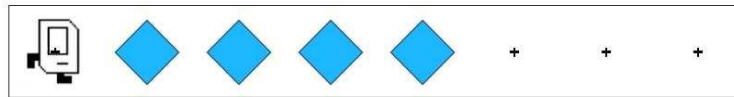
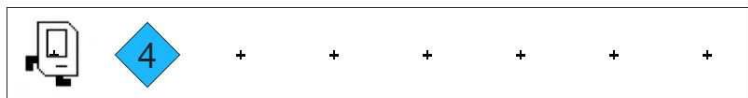
Algorithm 4



# Section Exercise: “Spread Beeper”

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

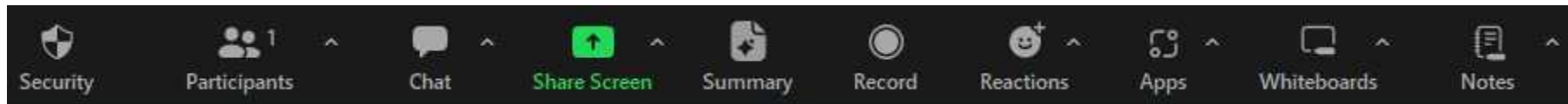
- This problem is intended to be harder than last week’s section problem!
- Outline problem logic
- Apply control flow to execute outline (while loops + conditionals)
- Multiple decompositions
- Think about the postcondition of loops, and if you are ready to repeat
  - Look for possible “fencepost errors”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



# Zoom: Breakout Rooms

Share your code!

- **Option 1:** Use the “**Share Screen**” button in Zoom’s bottom menu.
  - Recommend that you share your CIP browser window instead of your entire monitor screen.

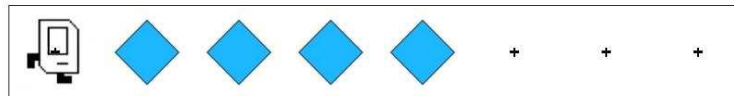
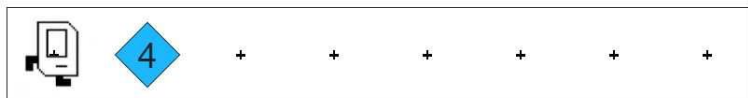


- **Option 2:** One person in your room goes to <https://codeshare.io/>
  - Click the “**Share Code Now**” button to create a new document.
  - Copy the URL from your browser window and paste it to the rest of your room. Everybody else go to that link. You can now type together.

# Section Exercise: “Spread Beeper”

Things to try in your breakout room:

- Start by coming up with a high-level strategy.
  - How would you explain your strategy to a 5 year old?
  - Remember decomposition. Break down a big problem into smaller easier problems.
  - Write out the pseudocode for your strategy.
  - Think about the helper functions you plan to implement.
    - Write out the pseudocode for the helper functions.
    - Remember pre-conditions & post-conditions.
- Hint: Think about how you might “spread” a single beeper first.



# Bonus Challenge: Multiple Rows!

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

