Welcome!

We'll get started shortly ...

Welcome to Section!

Week Two

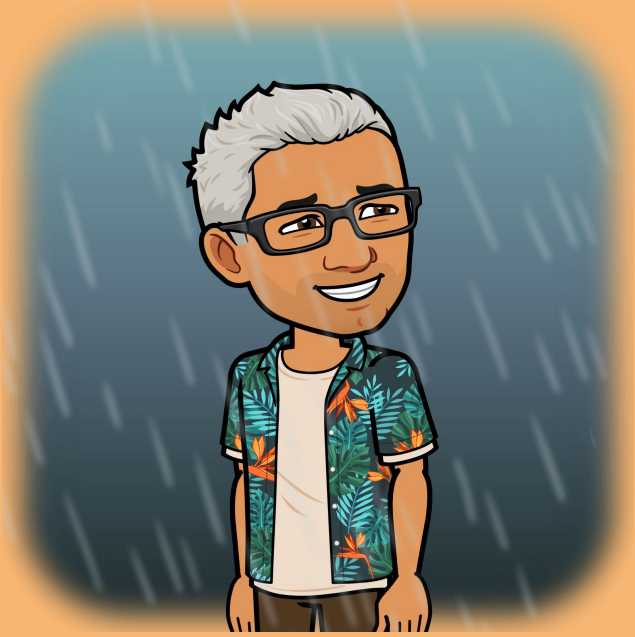
Cameron Mohne, Maggie Lee, Surajit Bose

Agenda

- Introductions and Logistics
- Review of lecture concepts
  - Control Flow: `for`, `while`, `if`, `if-else`
  - Functions
  - Worked Example
- Problem: Hospital Karel

Introductions

A Little About Me



- ★ My name is Surajit (he/him), and I'll be your Section Leader for CS 49
- ★ I am retired from a tech career
- ★ In my free time I like to read novels and poetry
- ★ I enjoy Indian classical music.

What About You All?

Please go ahead and share:

1. Your name, and if you would like to share them, your pronouns
2. Where you're tuning in from: locally from the Bay Area, or farther away?
3. What you'd like to get out of this class.

Another question: what can I do to make you feel included? Please feel free to private message me in the chat if there's anything I can do to make you more comfortable.

Breaking the Ice

- Share your names one more time!!!
- Icebreaker Question: What is your favorite home-cooked dish? Who makes it?
- If no one wants to share first, the person who is geographically closest to Stanford shares first!

How to get hold of me / get help+

- The section forum, 24 hr turnaround
- Email: bosesurajit@fhda.edu, 24 hr turnaround
- Office hours:
  - On campus: Tuesdays 12:00 noon to 1:30 pm, room 4218 in the STEM center. Entry is from room 4213
  - By appointment on Zoom
- Other resources:
  - Contact Lane via Canvas
  - Online or in-person tutoring via the STEM center (Room 4213)

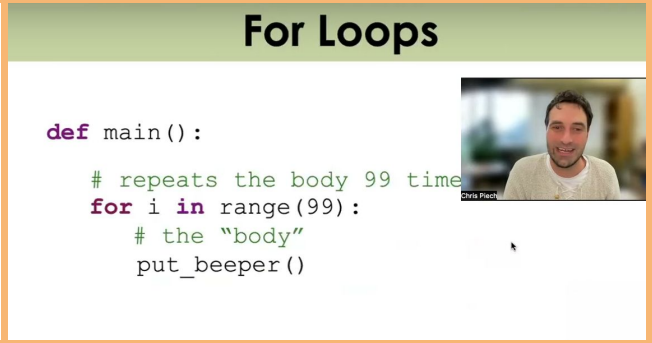
Lecture Review

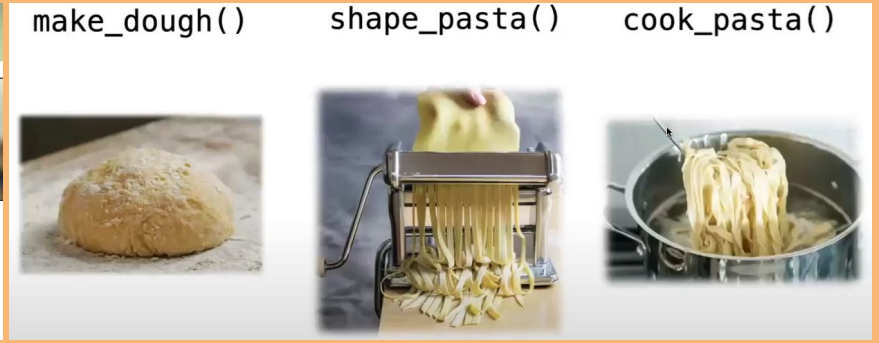
What We've Learned

Before we get into our sample problem for today, let's review a bit. We've learned:

- The basics about Karel, the magnificent and wonderful robot
- Control Flow, loops and conditional statements which guide our programs
- Functions, a way of breaking down big problems into smaller chunks

This is a LOT of content, especially if you are newer to CS!





Let's review and refresh these concepts a bit!

Control Flow Overview

`for` loop:

- Performs some block of code a specific number of times.

`while` loop:

- Continuously performs a block of code until a given condition is evaluated to false.

`if` statement:

- Performs a block of code only when a condition is true, and only once.

`if-else` statement:

- Performs a block of code when a condition is true, or a different block when the condition is false. Either block is performed only once.

`for` Loop

An example `for` loop that you may see and use with Karel:

```
def turn_right():
    for i in range(3):
        turn_left()
```

This loop is also called a *definite loop* because we know where it ends, when i reaches 3.

`while` Loop

An example `while` loop that you may see and use with Karel:

```
def move_to_wall():
    while front_is_clear():
        move()
```

This loop is also called an *indefinite loop* because it will run until the associated condition becomes false.

`if` Statement

An example `if` statement that you may see and use with Karel:

```
def safe_move():
    if front_is_clear():
        move()
```

An `if` statement runs code inside of it when the associated statement is evaluated to true.

`if-else` Statement

An `if-else` statement runs either the code inside the **`if`** block when the associated statement is evaluated to true, or the code inside the **`else`** block when the statement is evaluated to false. An example:

```
def safe_move_or_turn():
    if front_is_clear():
        move()
    else:
        turn_left()
```

We will get into more flexible statements later on in the course!

Any Questions?

Functions

Given a problem to solve:

- Start with the big picture
- Break the problem down into smaller, self-contained building blocks
  - These smaller building blocks are functions
  - The process of breaking down the problem into functions is **decomposition**
  - Any set of steps that will need to be repeated is a good candidate for a function
- Assume the building blocks are done (use `pass` keyword)
- Assemble building in `main()` to solve the big problem
- Implement each building block!

Functions

Example: Karel needs to walk from the first corner `[1,1]` to the end of the row. Every time Karel lands on a beeper, it needs to spin 360º .

Big picture:

- Karel starts on `[1,1]`
- It moves until it is on a corner with a beeper.
- It spins, then moves forward again.
- This process continues until Karel reaches a wall.

What is the small building block that will be useful?

- What action does Karel not yet know how to do, but will need to do repeatedly?

Functions

```
spin()
```

Any Questions?

Worked Example

Using functions and control flow to make Karel spin when it is on a beeper

https://codeinplace.stanford.edu/foothill-cs49/ide/p/UWYFIAF12e6QZxdXoQg6

Solve the big problem

```
def main():
    while front_is_clear():
        move()
        if beepers_present():
            spin()

def spin():
    pass     # Placeholder
```

Implement the building blocks

```
def main():
    while front_is_clear():
        move()
        if beepers_present():
            spin()

def spin():
    for i in range(4):
        turn_left()
```

Let's Try It Out!

Did it work?

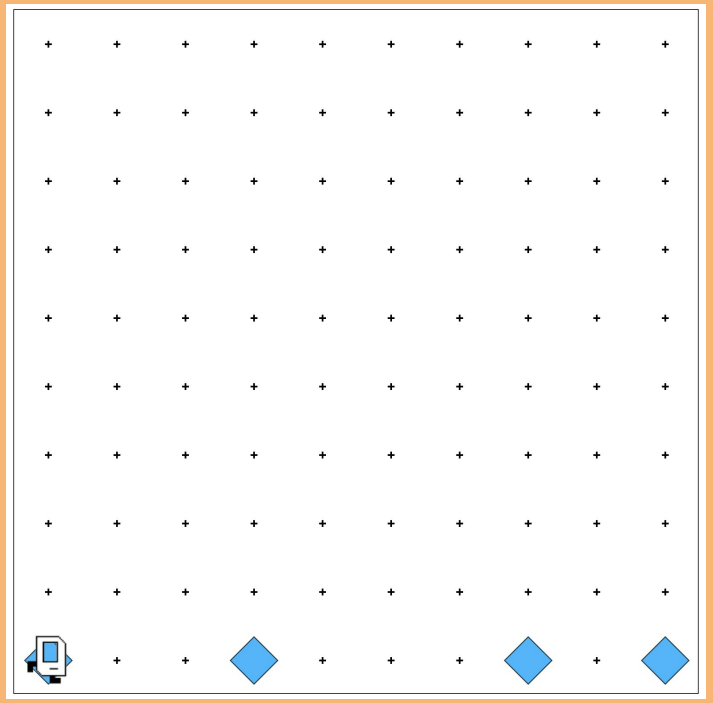
Test and refine the entire solution

```
def main():
    if beepers_present():      #  Fencepost problem: there
        spin()                 #  could be a beeper at [1,1]
    while front_is_clear():
        move()
        if beepers_present():
            spin()

def spin():
    for i in range(4):
        turn_left()
```

`spin()` in action

You can see Karel spinning as desired here. The implementation uses the code on the previous slide.



Any Questions?

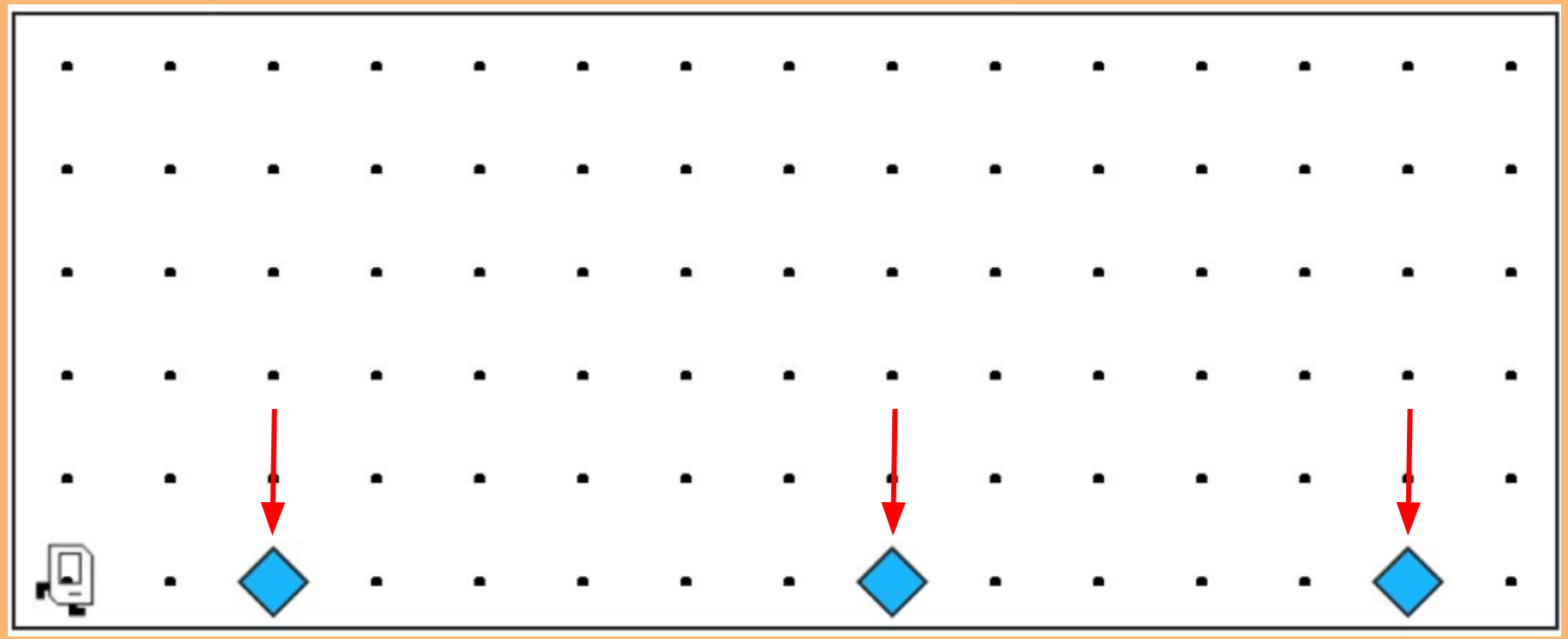
Section Problem: Hospital Karel

https://codeinplace.stanford.edu/foothill-cs49/ide/a/hospital

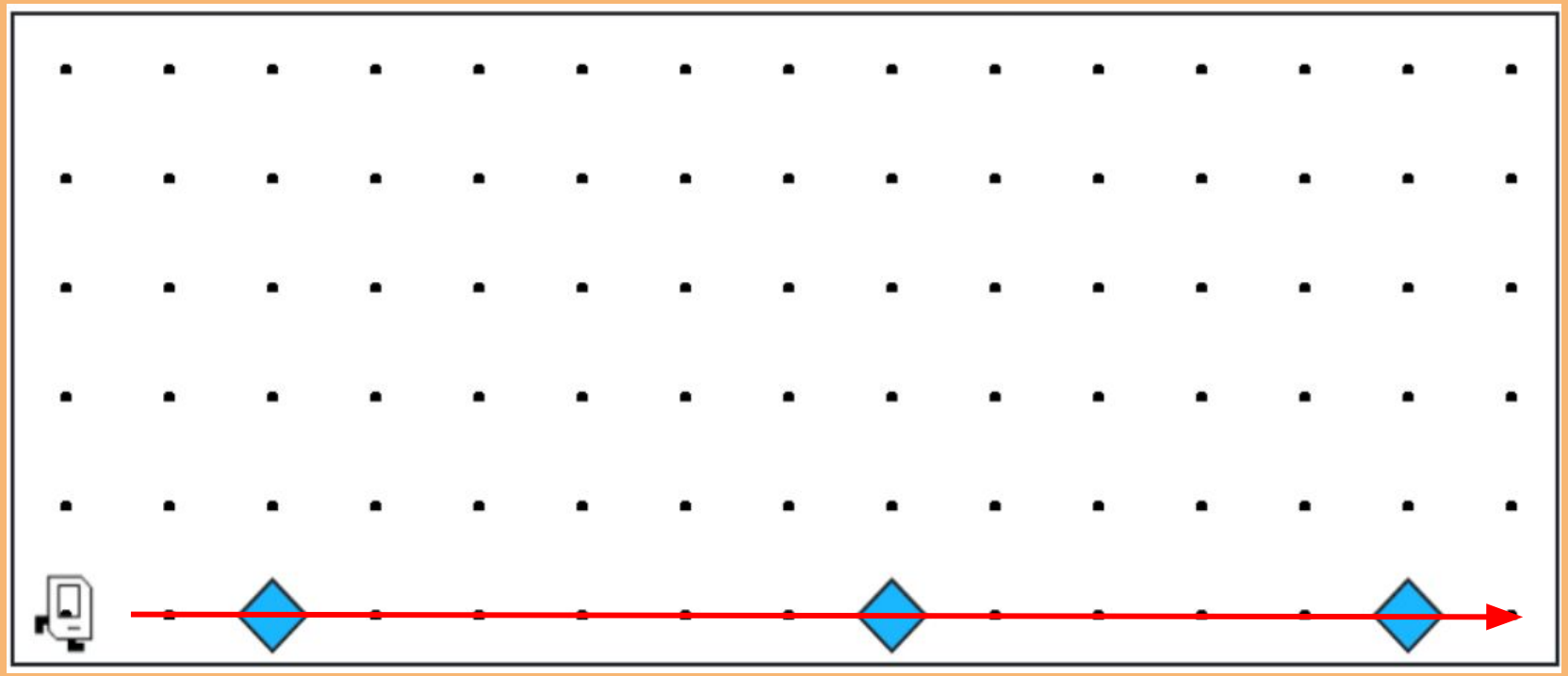
Setting Context

Countries around the world are dispatching hospital-building robots to make sure anyone who gets sick can be treated. They have decided to enlist Karel robots. Your job is to program those robots.

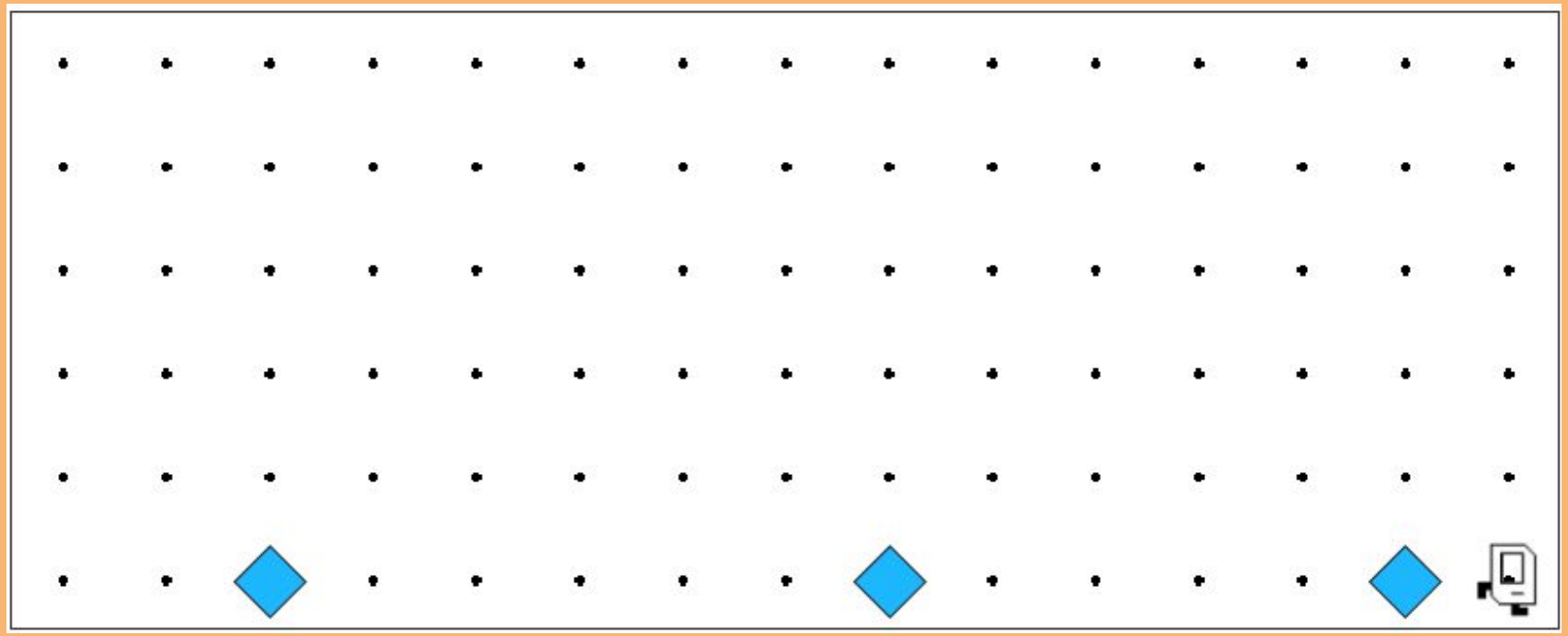
Each beeper in the figure represents a pile of supplies.



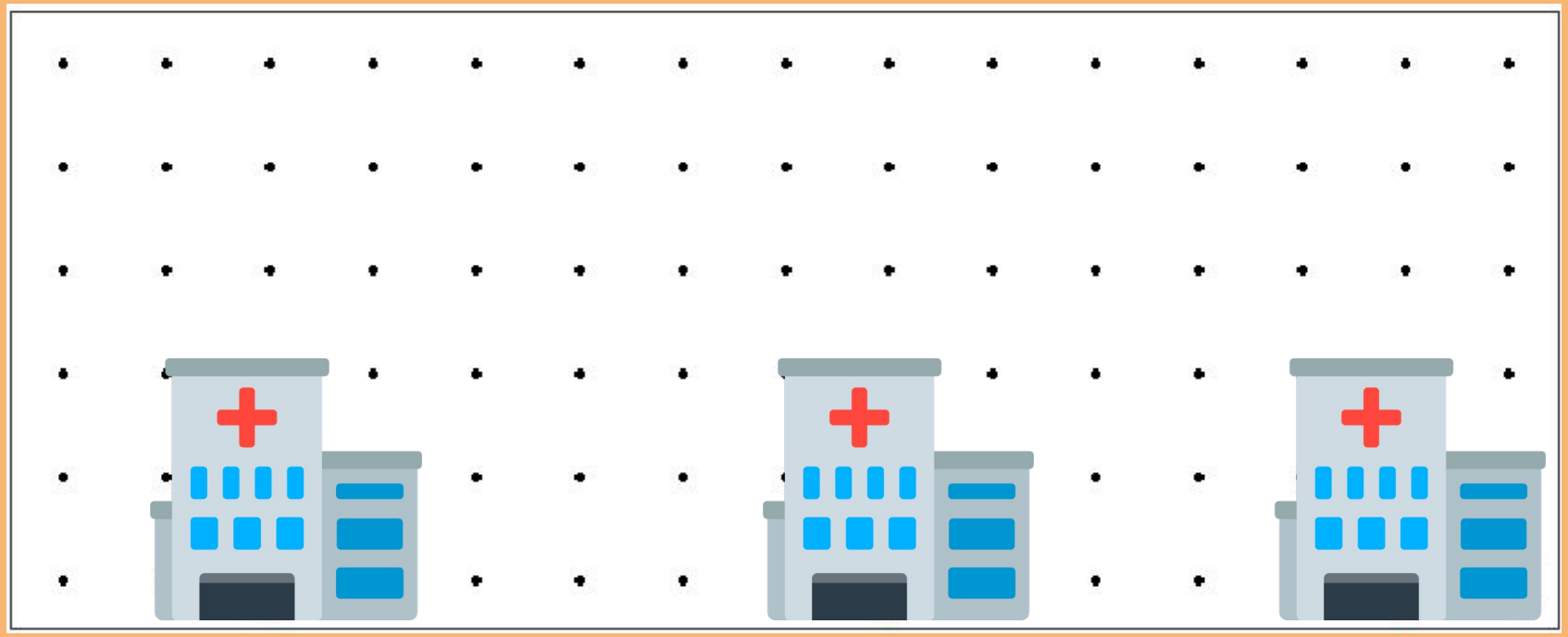
Karel's job is to walk along the row and build a new hospital in the places marked by each beeper.



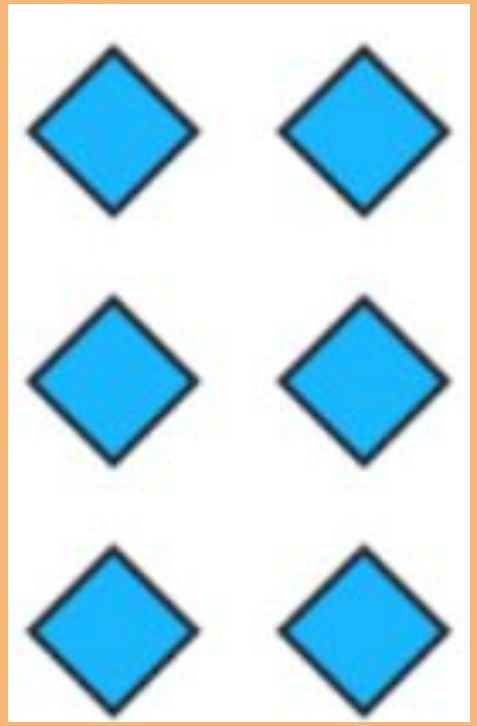
Karel's job is to walk along the row and build a new hospital in the places marked by each beeper.



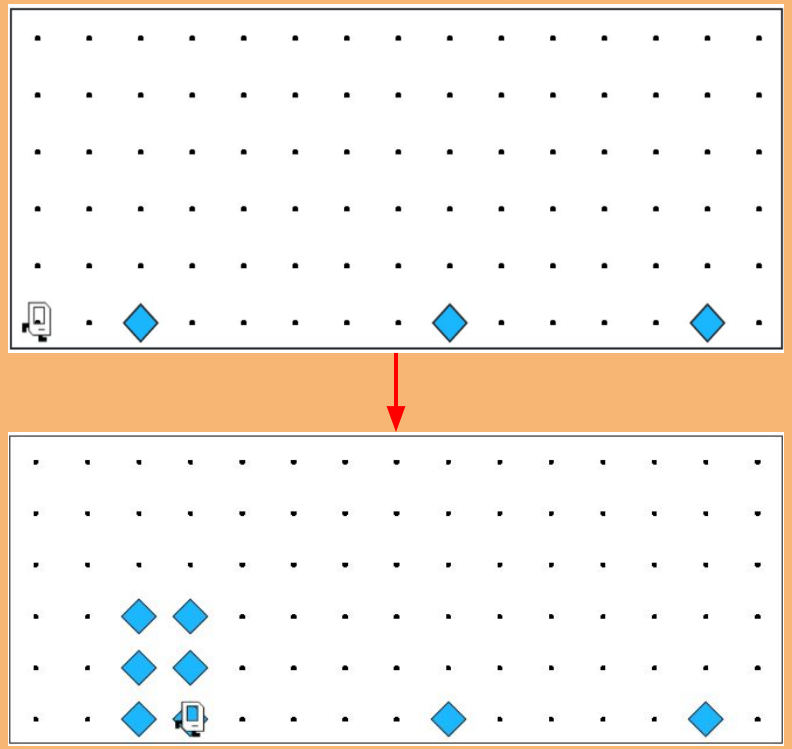
Karel's job is to walk along the row and build a new hospital in the places marked by each beeper.



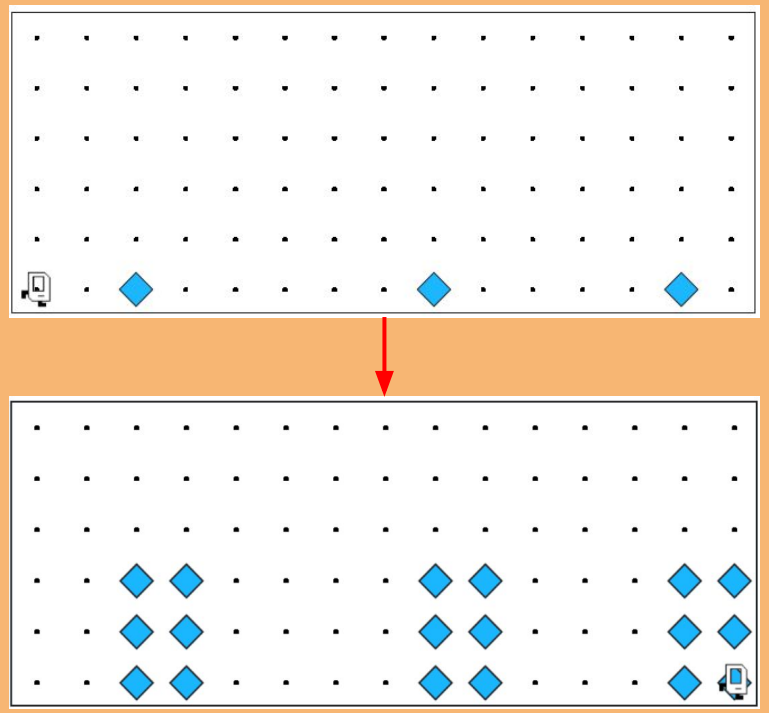
Hospitals look like this: a 3x2 rectangle of beepers!



The new hospital should have their first corner at the point at which the pile of supplies was left.



At the end of the run, Karel should be at the end of the row having created a set of hospitals. For the initial conditions shown, the result would look like this:



Notes to Keep in Mind

- Karel starts facing east at `[1, 1]` with an infinite number of beepers in its beeper bag
- The beepers indicating the positions at which hospitals should be built will be spaced so that there is room to build the hospitals without overlapping or hitting walls
- There will be no supplies left on the last column
- Karel should not run into a wall if it builds a hospital that extends into that final corner.

Questions Before We Begin?

Let's get to work!