Welcome to Section!!!

Week One

Congratulations

A Little About Me

- ★ My name is **Mah Noor Fatima** . I'll be your Section Leader for **Code in Place** , and everyone here will be your mates!
- ★ I am working in Software Engineering.
- ★ In my free time I like to Read Books.
- ★ I enjoy Music & Coffee.

What About You All?

Go ahead and share:

1. **Your name**
2. **Where you're turning in from**
3. **One thing you're looking forward to (it doesn't have to be from Code in Place)!**

Breaking the Ice

In breakout rooms:

- Share your names one more time!!!
- Icebreaker Question: What's your dream project?
- If no one wants to share first, the person who is geographically closest to Stanford shares first!

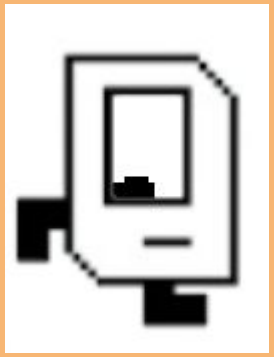
Lecture Review

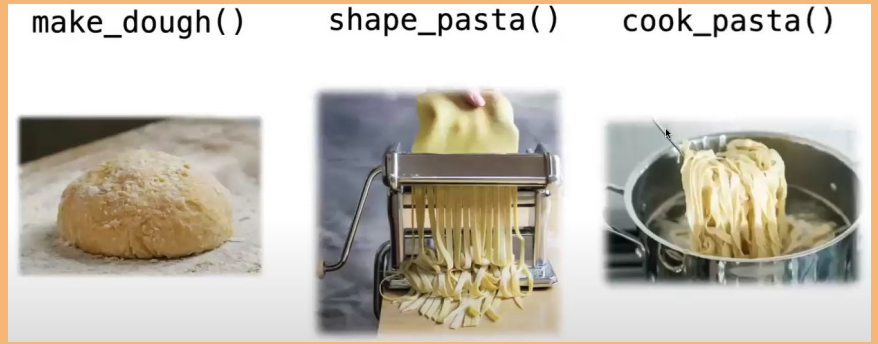
What We've Learned

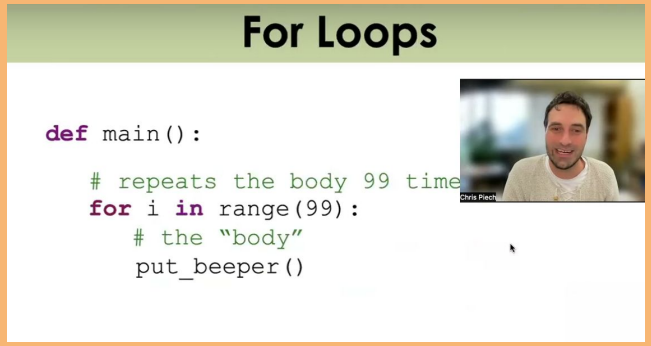
Before we get into our sample problem for today, let's review a bit. We've learned:

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

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







Let's review and refresh these concepts a bit!

Karel Overview

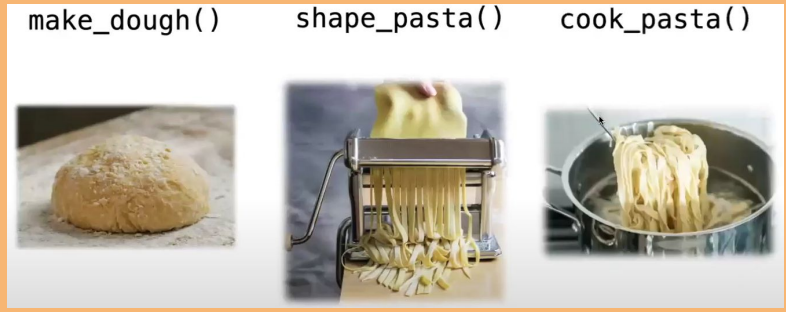
- Karel is a small, but mighty robot!!!
- It has a few basic commands including: **`move()`**, **`turn_left()`**, **`pick_beeper()`**, and **`put_beeper()`**
- On its own, Karel has limited functionality, but with the help of our code, we can make great things happen!

Functions Overview

When you think of functions, recall Chris and Mehran's analogy to **making pasta**. Each function has a specific purpose which breaks down a larger problem into smaller chunks—just like steps in making pasta from scratch!

To make a function, you need to define it using the def keyword. Afterwords, write the code you want the function to run. Make sure the code is indented below the function name like so:

```
def function_name():
    # Function code goes here!!!
```

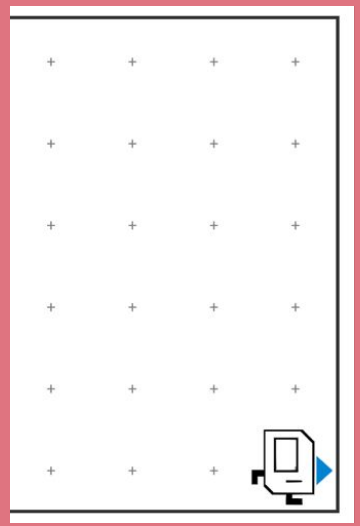


Control flow

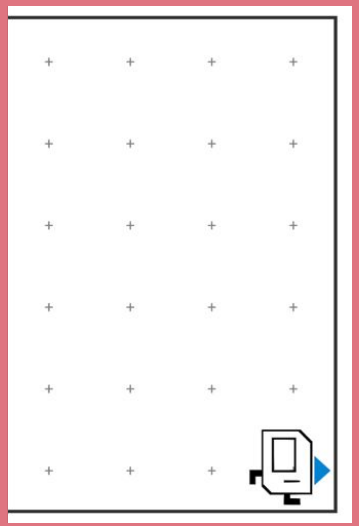
Control flow

- `if`
- `if/else`
- `for`
- `while`

Do something when a condition is met.



```
if front_is_clear():
  move()
```



Control flow

Control flow

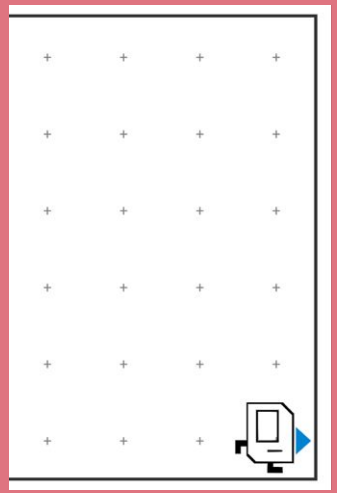
`if`

`if/else`

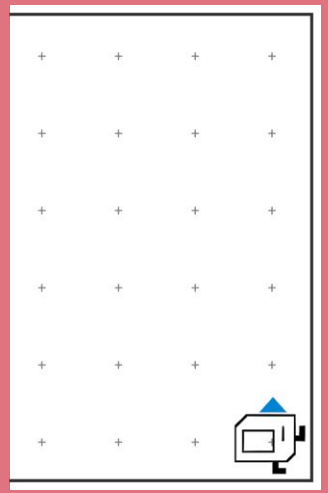
`for`

`while`

Also do something else
when a condition is not met.



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



Control flow

Control flow

`if`

`if/else`

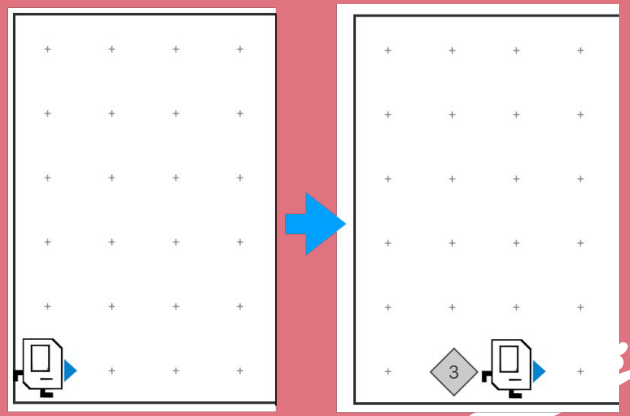
`for`

`while`

Move forward 8 squares

```
for i in range(8):
```

Do something a set number of times.



Place 3 beepers

```
for i in range(3):
```

Control flow

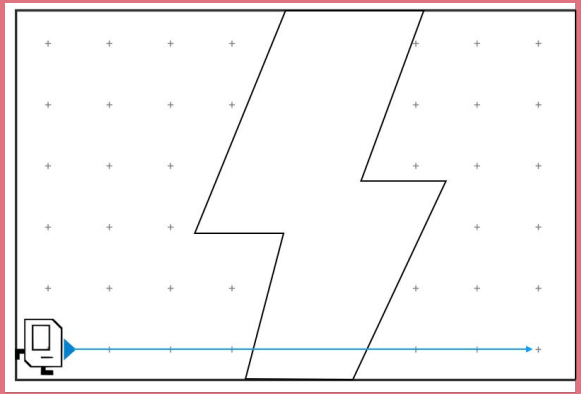
Control flow

`if`

`if/else`

`for`

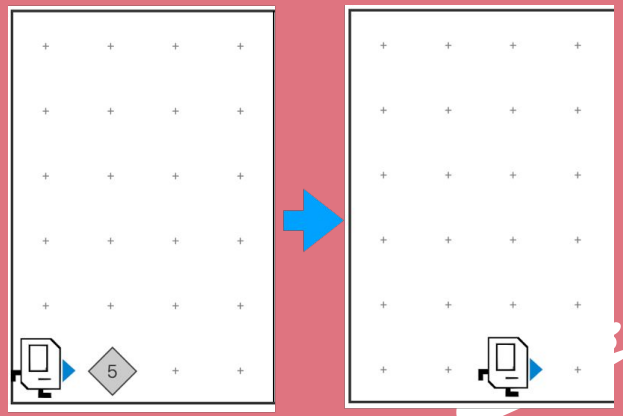
`while`



Move forward as long as the front is clear

```
while front_is_clear():
```

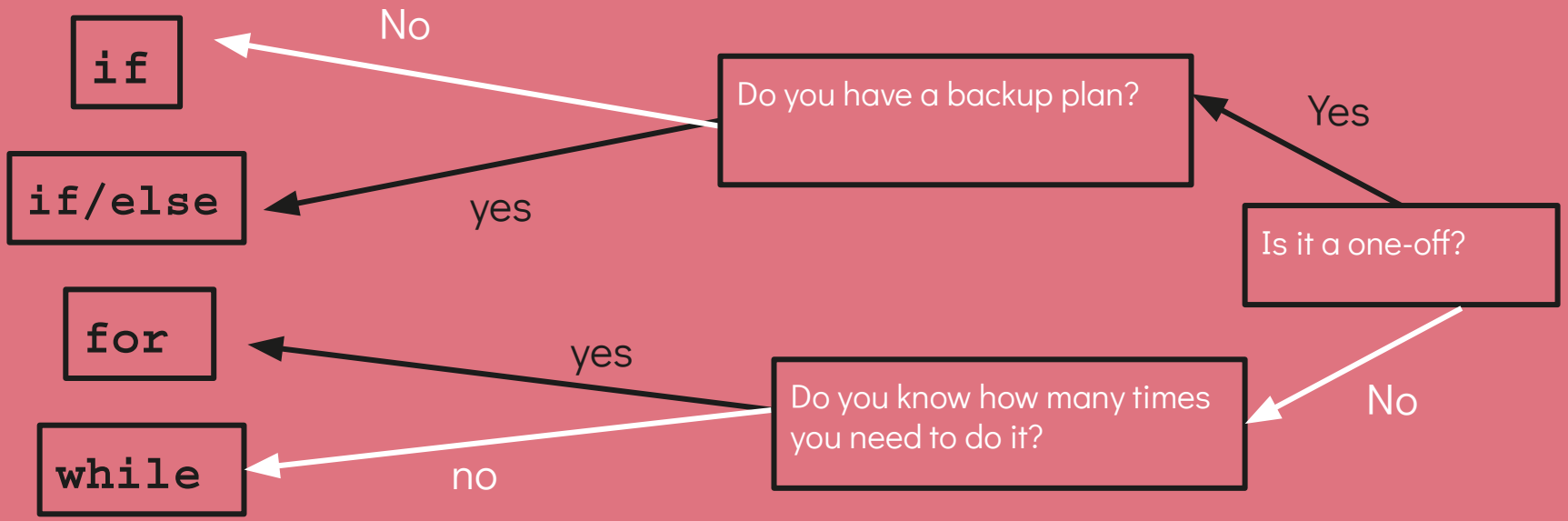
Do something until condition is no longer true.



Pick up beepers while there are still some to pick up

```
while beepers_present():
```

Control flow



Decomposition

Decomposition

Define your function like this:

```
def function_name():
    <write code here>
```

And call it like this:

```
function_name()
```

Decomposition

Decomposition

More of an art than a science.

Functions should be short and read like English.

If you repeat things (or find yourself hitting copy/paste), take a step back.

Top-down programming & the leap of faith.

Decomposition

Decomposition

```
def main():
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
```

Decomposition

Decomposition

```
def main():
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
```

```
def main():
    spin()
    spin()
    spin()

def spin():
    turn_left()
    turn_left()
    turn_left()
    turn_left()
```

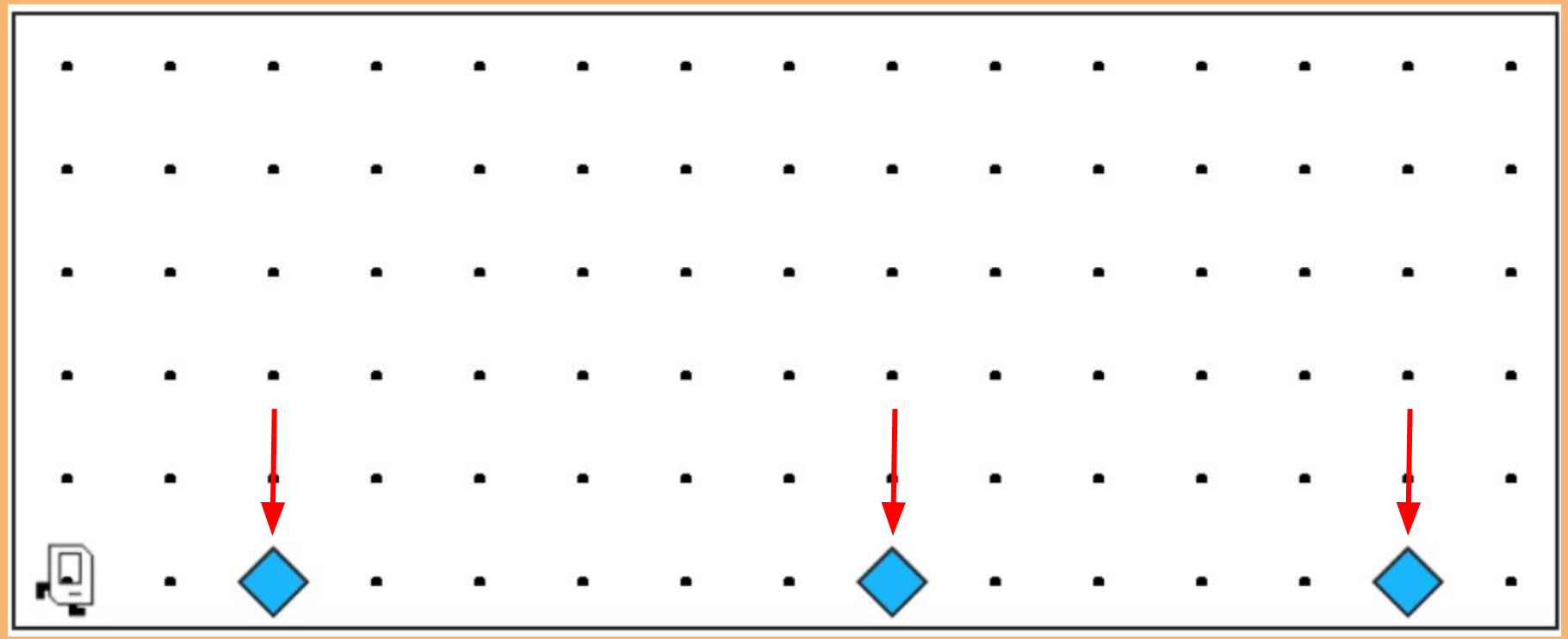
Any Questions?

Section Problem: Hospital Karel

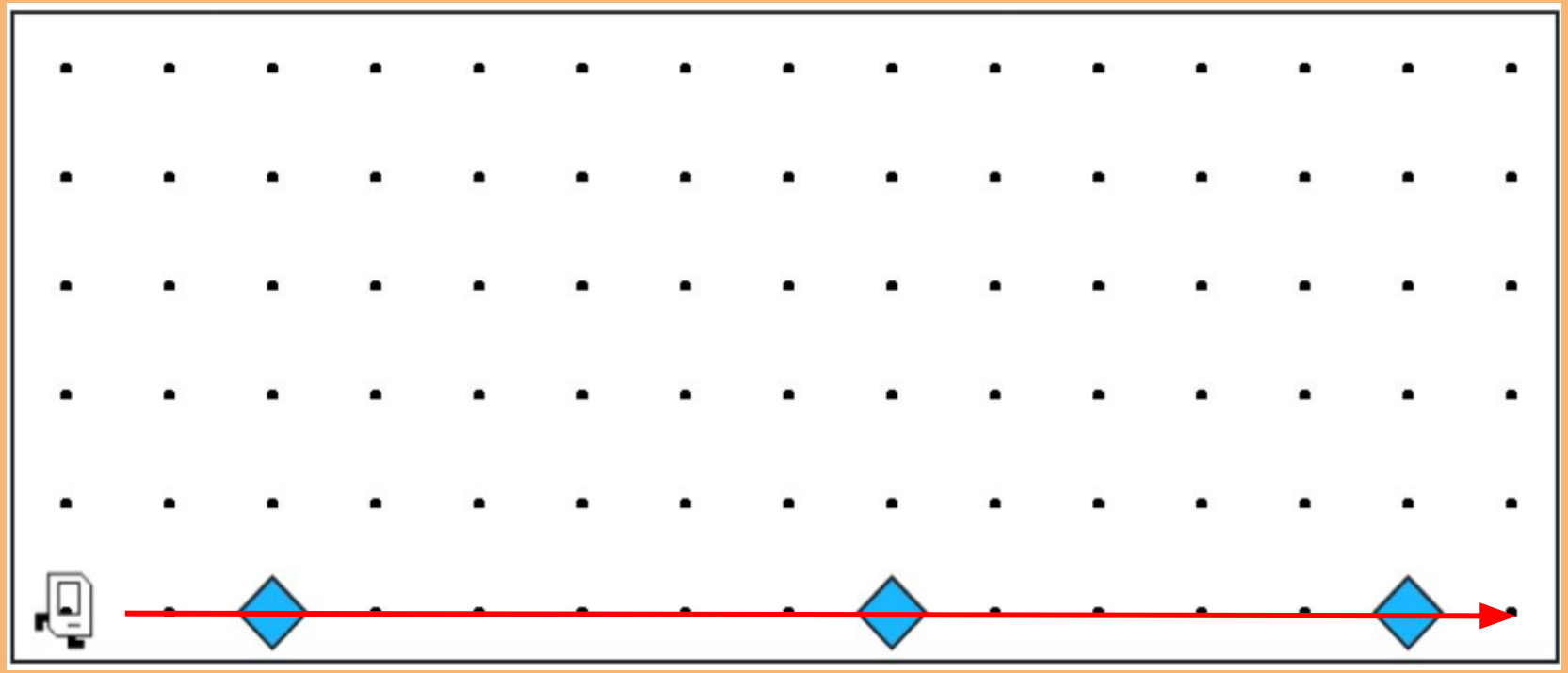
Setting Context

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.

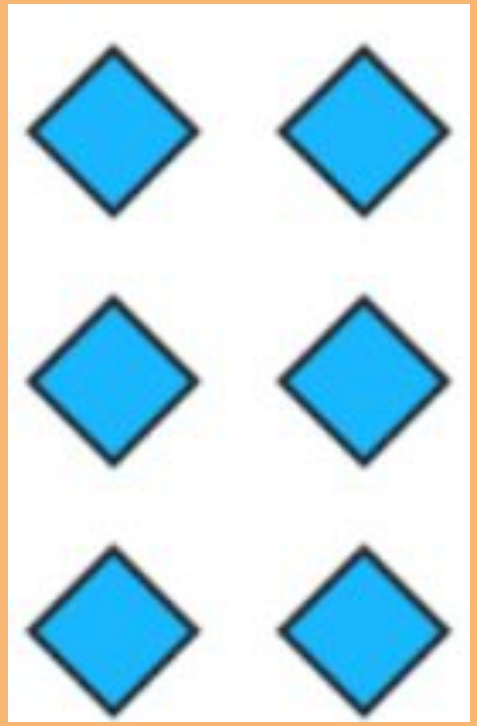


Karel's job is to walk along the row and build a new hospital in the places marked by each beeper.

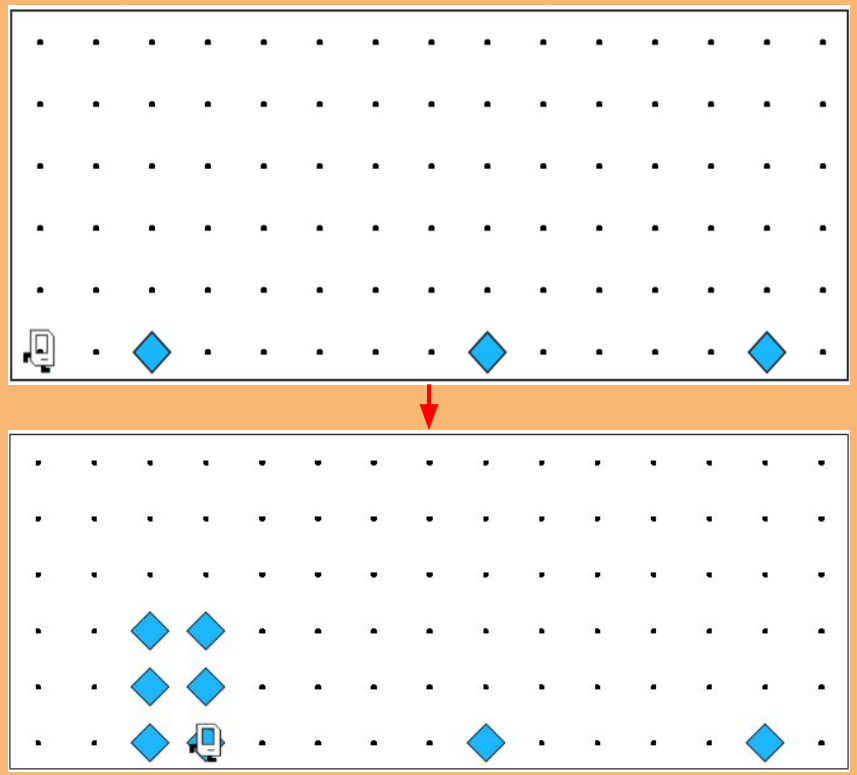


Karel's job is to walk along the row and build a new hospital in the places marked by each beeper.

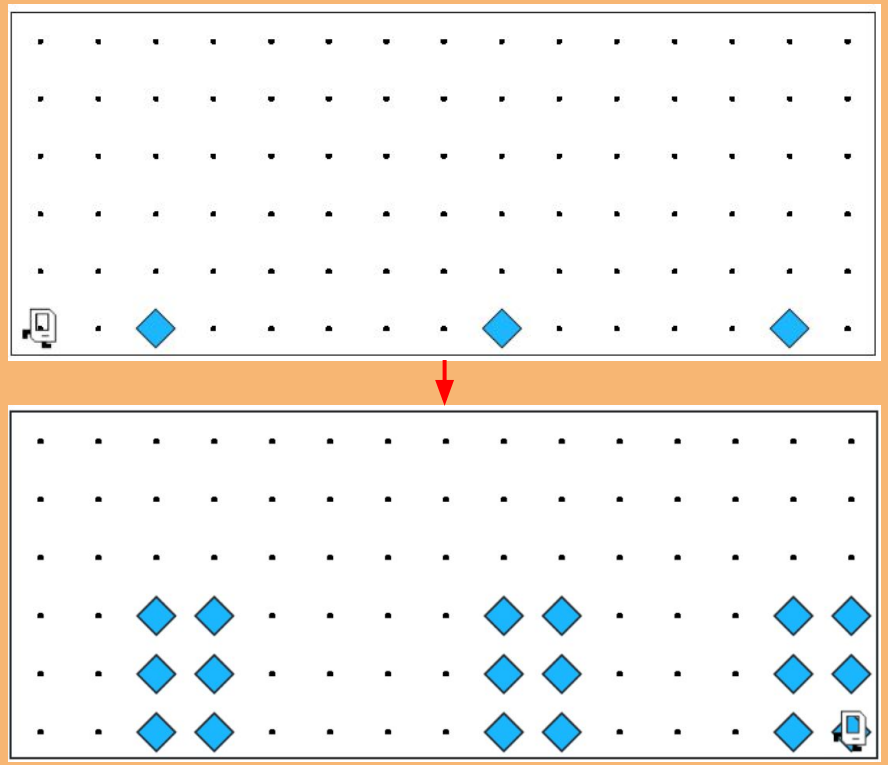
Hospitals look like this: a 3x2 rectangle of beepers!



The new hospital should have their 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!