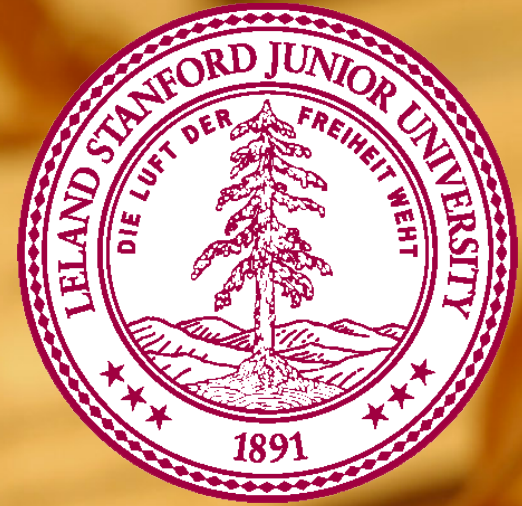




Decomposition

Chris Piech
CS106A, Stanford University

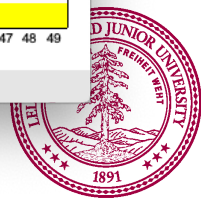
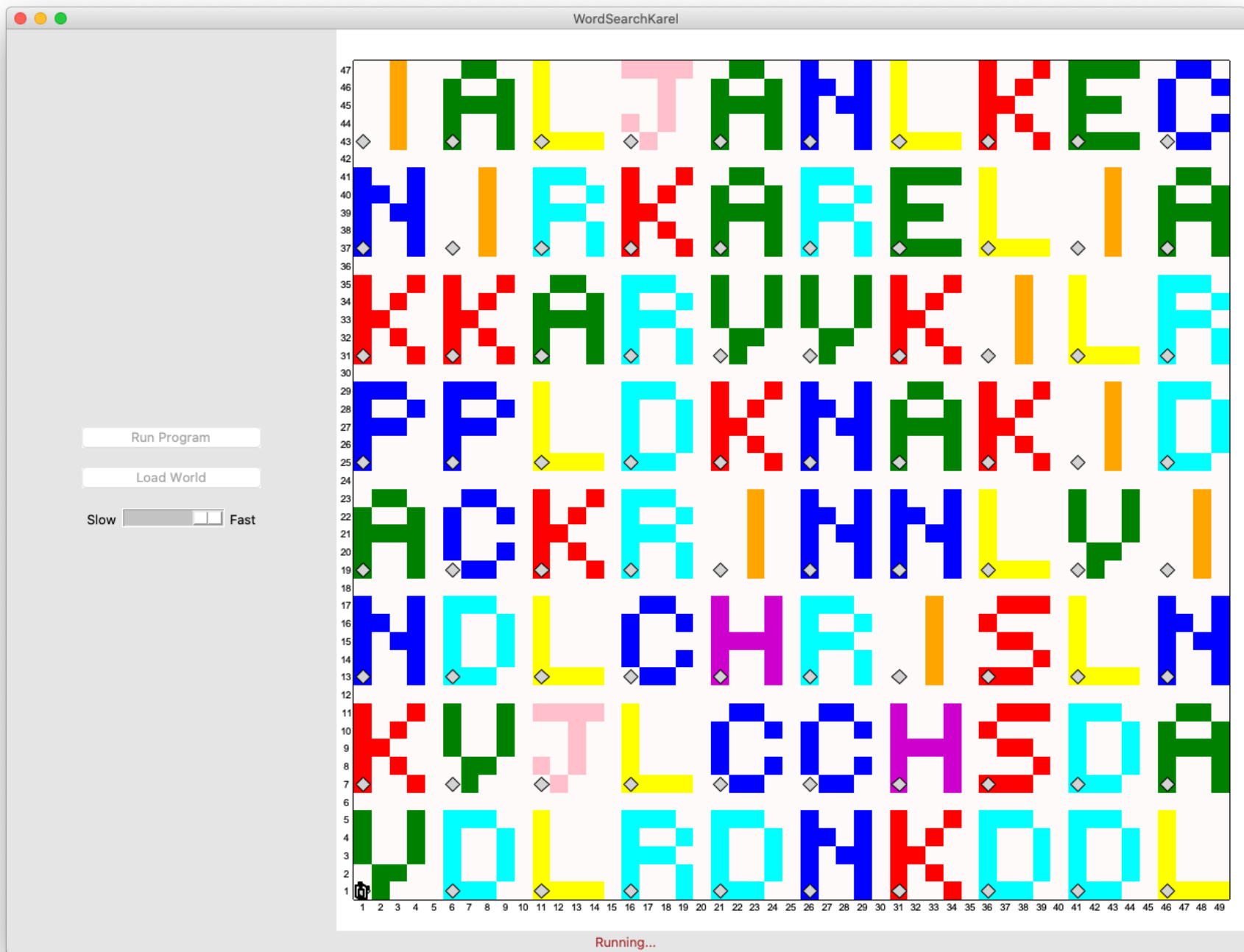
Today's Goal

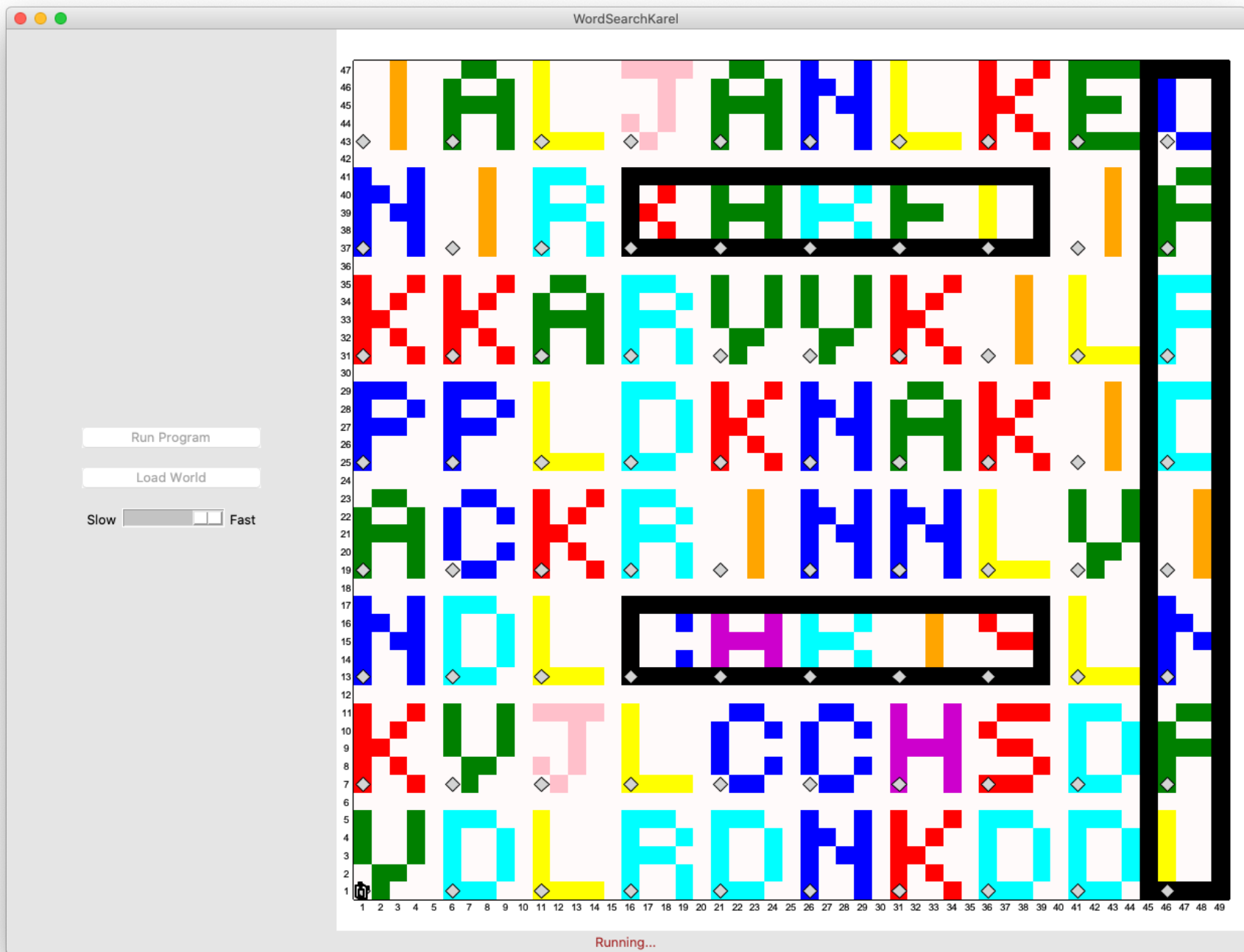
1. Be able to approach a problem “top down” by using decomposition (aka top town refinement)



First, a cool program

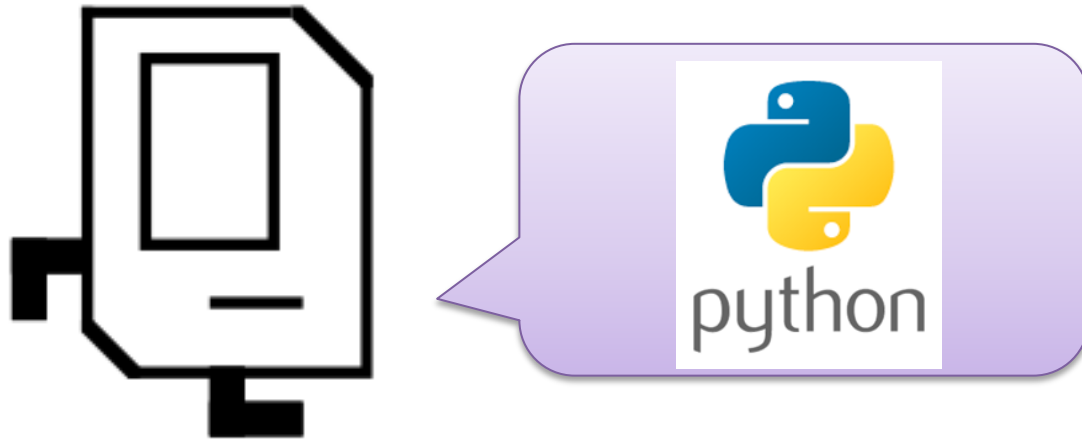






Quick review

Karel the Robot



Functions

```
def main():  
    goToMoon()
```

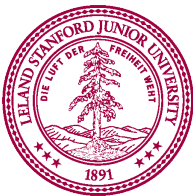
```
def go_to_moon():  
    build_spaceship()  
    # a few more steps
```

```
def build_spaceship():  
    # todo  
    put_beeper()
```



For Loops

```
def main():  
    # repeats the body 99 times  
    for i in range(99):  
        # the "body"  
        put_beeper()
```



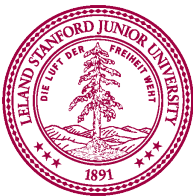
While Loops

```
def main():  
  
    # while condition holds runs body  
    # checks condition after body completes  
    while front_is_clear():  
        move()
```



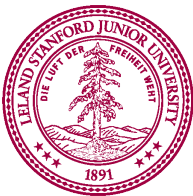
If Statement

```
def main():  
  
    # If the condition holds, runs body  
    if front_is_clear():  
        move()
```



If / Else Statement

```
def main():  
  
    # If the condition holds,  
    if beepers_present():  
        # do this  
        pick_beeper()  
    else :  
        # otherwise, do this  
        put_beeper()
```



The Full Karel

Base Karel commnds: move() turn_left() put_beeper() pick_beeper()	Conditions: if <i>condition</i> : <i>code run if condition passes</i> if <i>condition</i> : <i>code block for "yes"</i> else: <i>code block for "no"</i>
Karel program structures: # Comments can be included in any part # of a program. They start with a # # and include the rest of the line. def main() : <i>code to execute</i> <i>declarations of other functions</i>	Loops: for i in range(<i>count</i>): <i>code to repeat</i> while <i>condition</i> : <i>code to repeat</i>
Names of the conditions: front_is_clear() front_is_blocked() beepers_present() no_beepers_present() beepers_in_bag() no_beepers_in_bag() left_is_clear() left_is_blocked() right_is_clear() right_is_blocked() facing_north() not_facing_north() facing_south() not_facing_south() facing_east() not_facing_east() facing_west() not_facing_west()	Function Declaration: def <i>name</i> (): <i>code in the body of the function.</i> Extra Karel Commands: paint_corner(<i>COLOR_NAME</i>) corner_color_is(<i>COLOR_NAME</i>)



End review

```
def friday():  
    # heres our plan  
    decomposition()  
    double_beeper()  
    rhomba_karel()  
    if extra_time():  
        word_search_karel()
```

```
def friday():  
    # heres our plan  
    decomposition()  
    double_beeper()  
    rhomba_karel()  
    if extra_time():  
        word_search_karel()
```

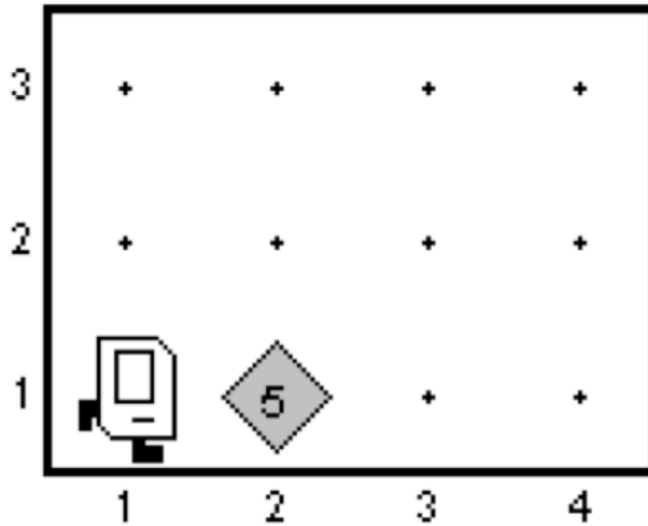


```
def friday():  
    # heres our plan  
    decomposition()  
    double_beeper()  
    rhomba_karel()  
    if extra_time():  
        word_search_karel()
```

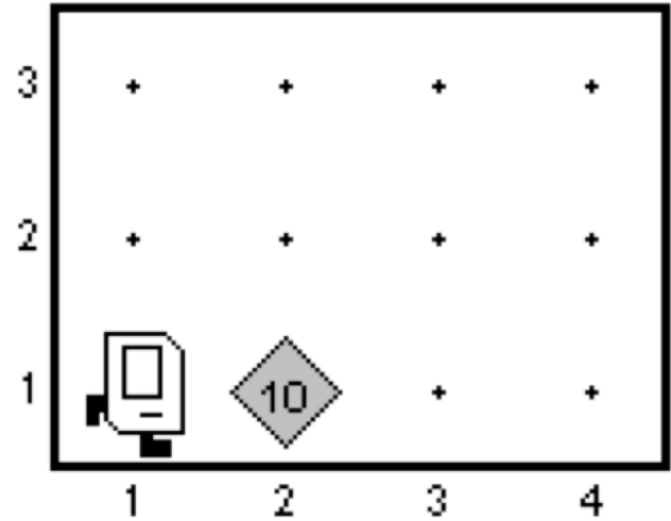
```
def friday():  
    # heres our plan  
    decomposition()  
    double_beeper()  
    rhoomba_karel()  
    if extra_time():  
        word_search_karel()
```

Double Beepers

Before



After



Muhammed ibn
Musa Al Kwarizmi

DO
YOUR
THING



Aside: Common Errors

Lather,
Rinse,
Repeat

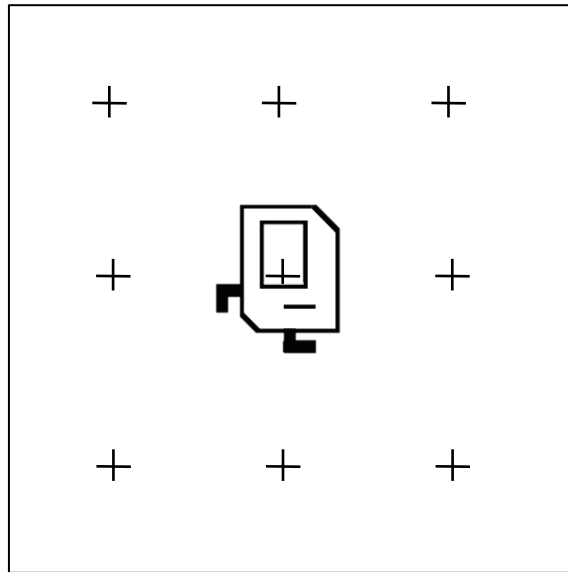
Now that your
longer,
ed
Balsam.

conditions
ing healthy
ch easier to
oo. You just
hower after
sure you get
only Wella makes
the original Balsam, and it's
great stuff. Wella Balsam.



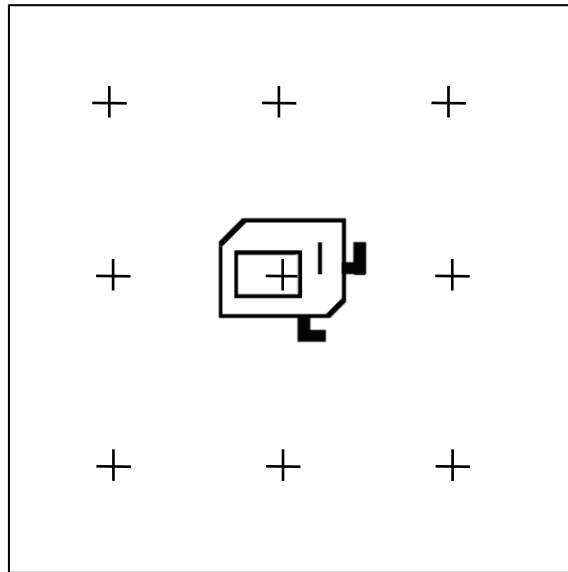
Infinite Loop

```
def turn_to_wall():  
    while left_is_clear():  
        turn_left()
```



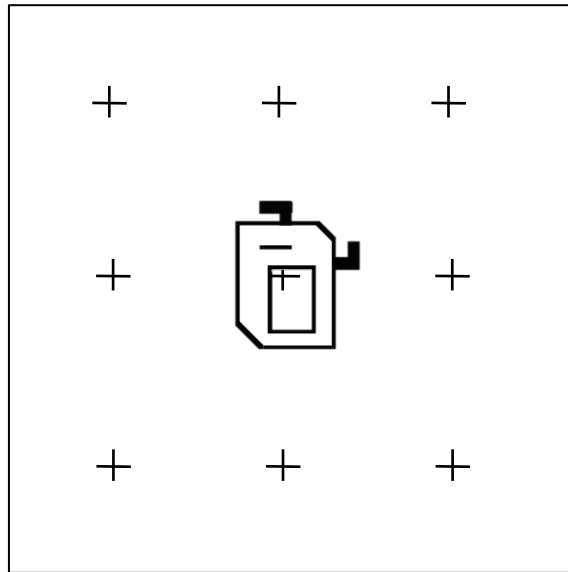
Infinite Loop

```
def turn_to_wall():  
    while left_is_clear():  
        turn_left()
```



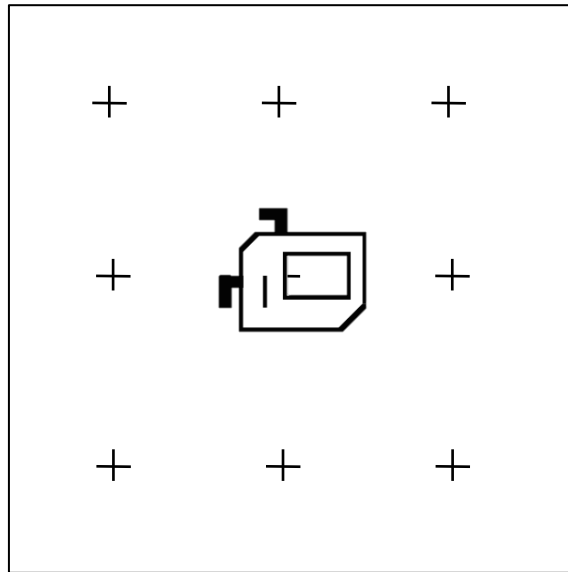
Infinite Loop

```
def turn_to_wall():  
    while left_is_clear():  
        turn_left()
```



Infinite Loop

```
def turn_to_wall():  
    while left_is_clear():  
        turn_left()
```

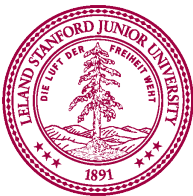


Pre/Post that Don't Match

```
def jump_hurdles:
    for i range(8):
        if front_is_clear():
            move()
        else:
            jump_hurdle()
```

What do you
assume here?

Does the “post
condition” match?

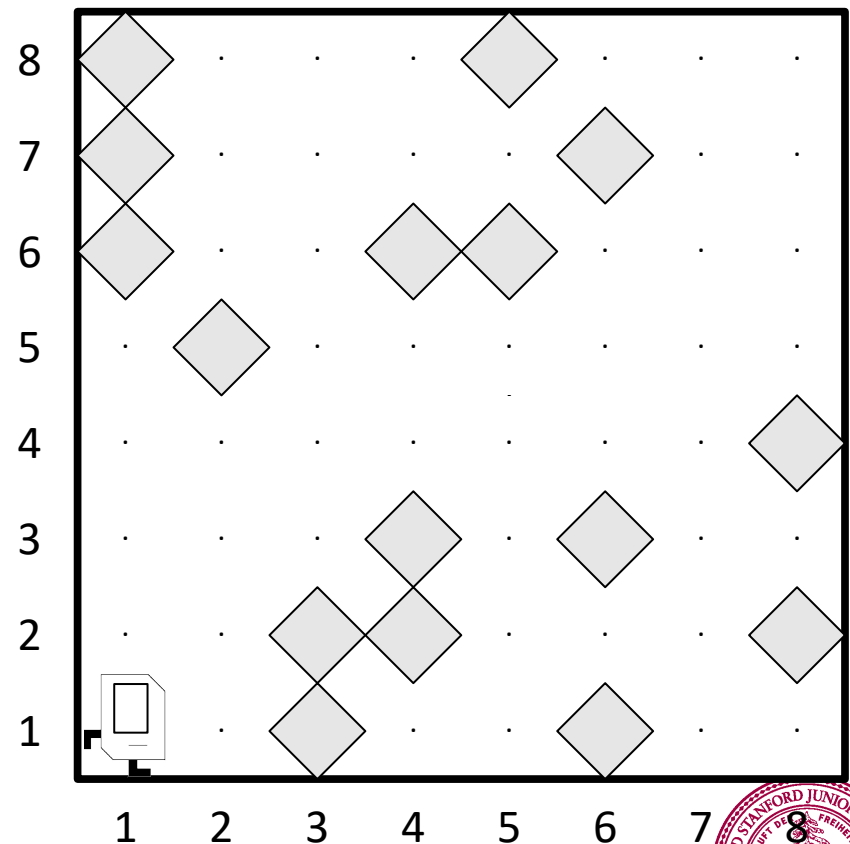


```
def friday():  
    # heres our plan  
    decomposition()  
    double_beeper()  
    rhoomba_karel()  
    if extra_time():  
        word_search_karel()
```

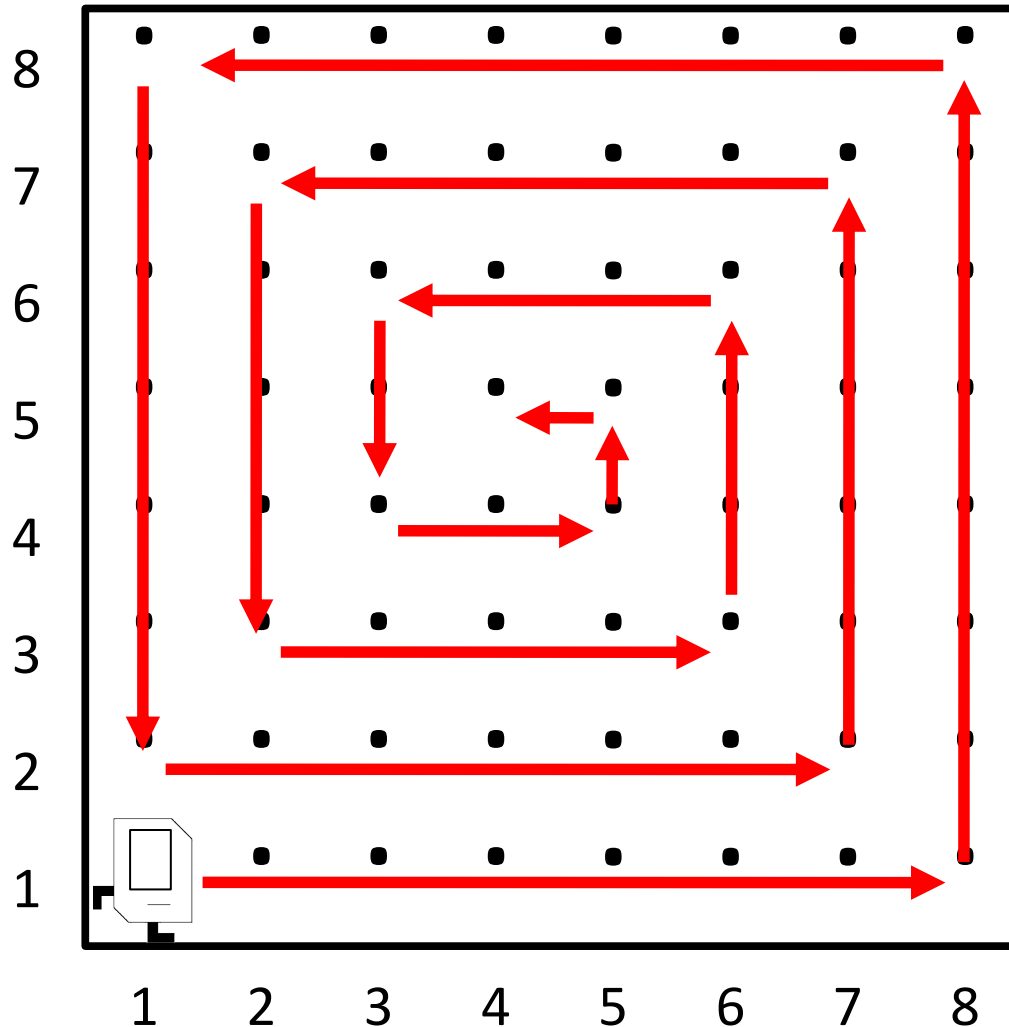
```
def friday():  
    # heres our plan  
    decomposition()  
    double_beeper()  
    rhomba_karel()  
    if extra_time():  
        word_search_karel()
```

Rhomba Karel

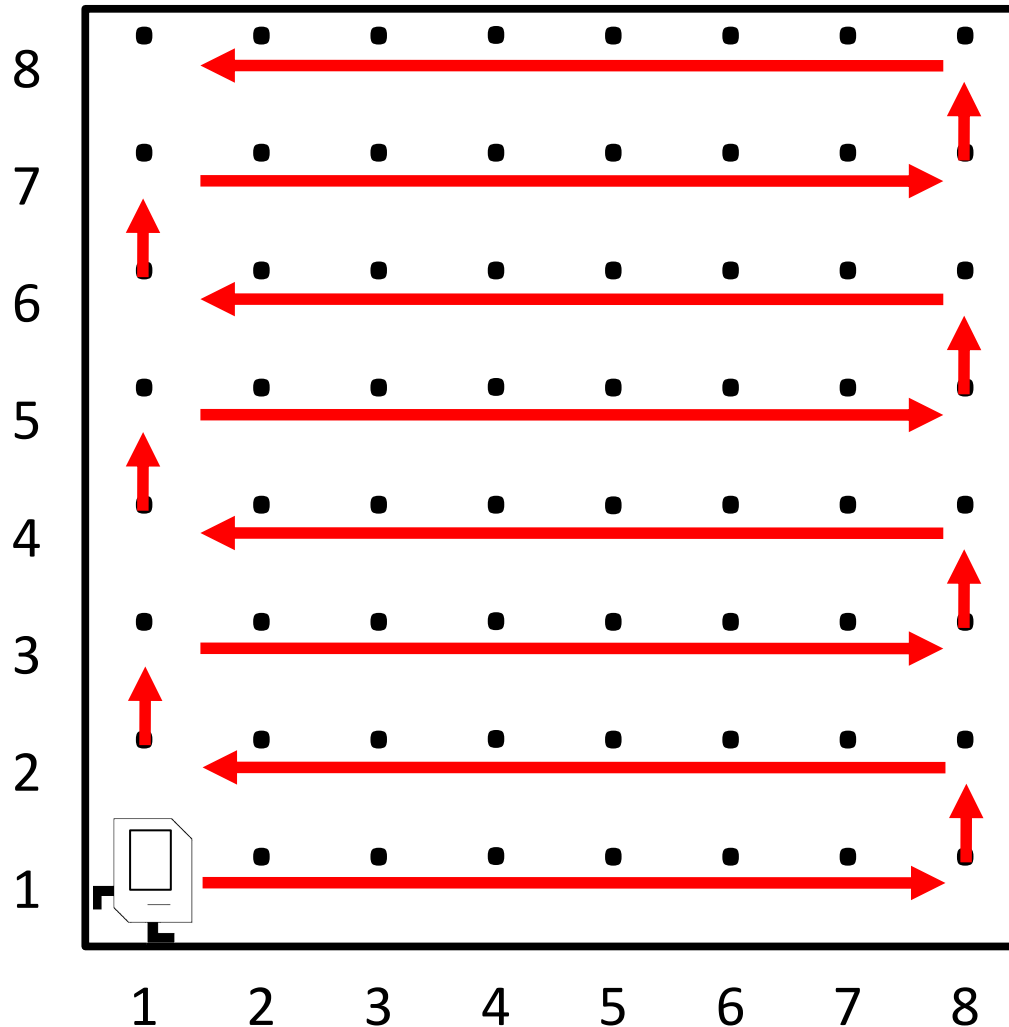
- Write a **Roomba** Karel that sweeps the entire world of all beepers.
 - Karel starts at (1,1) facing East.
 - The world is rectangular, and some squares contain beepers.
 - There are no interior walls.
 - When the program is done, the world should contain 0 beepers.
 - Karel's ending location does not matter.
- How should we approach this tricky problem?



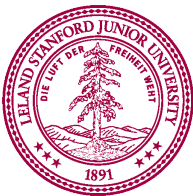
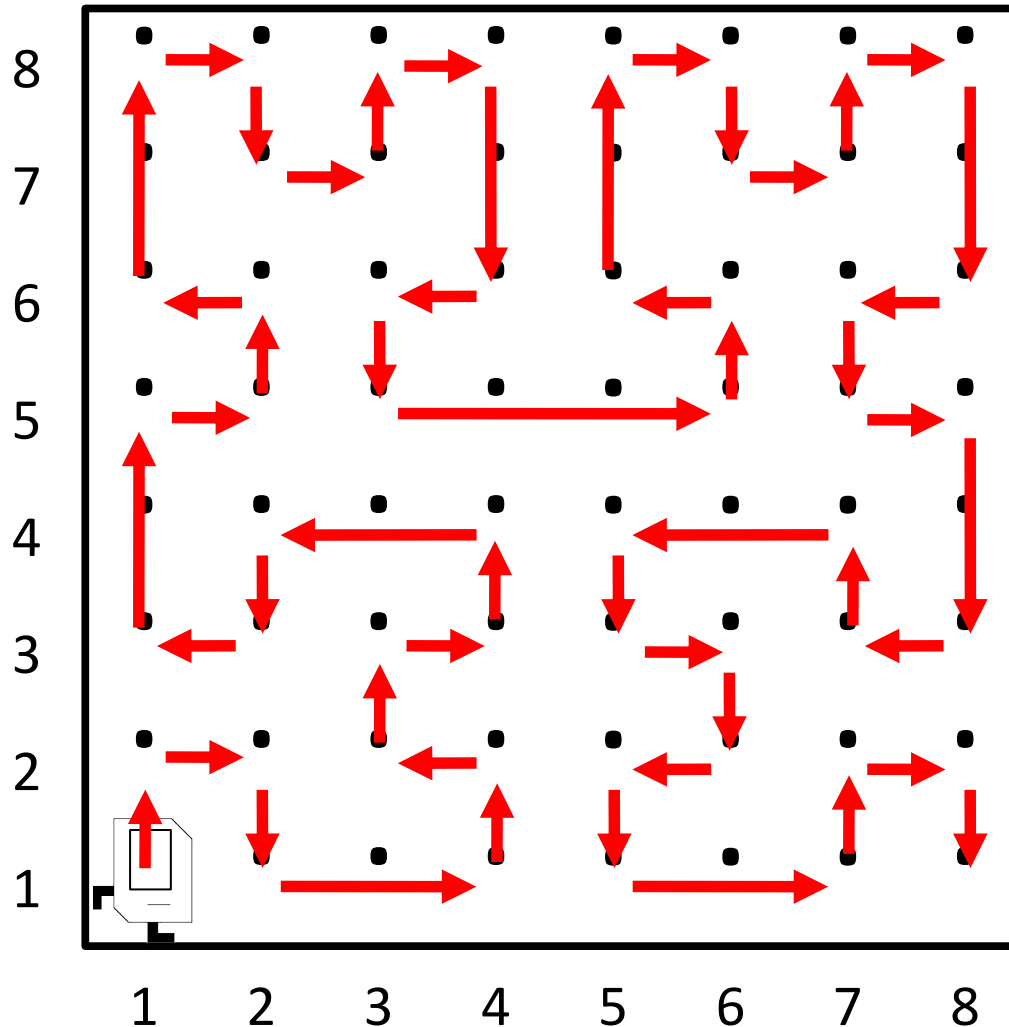
Possible Algorithm 1



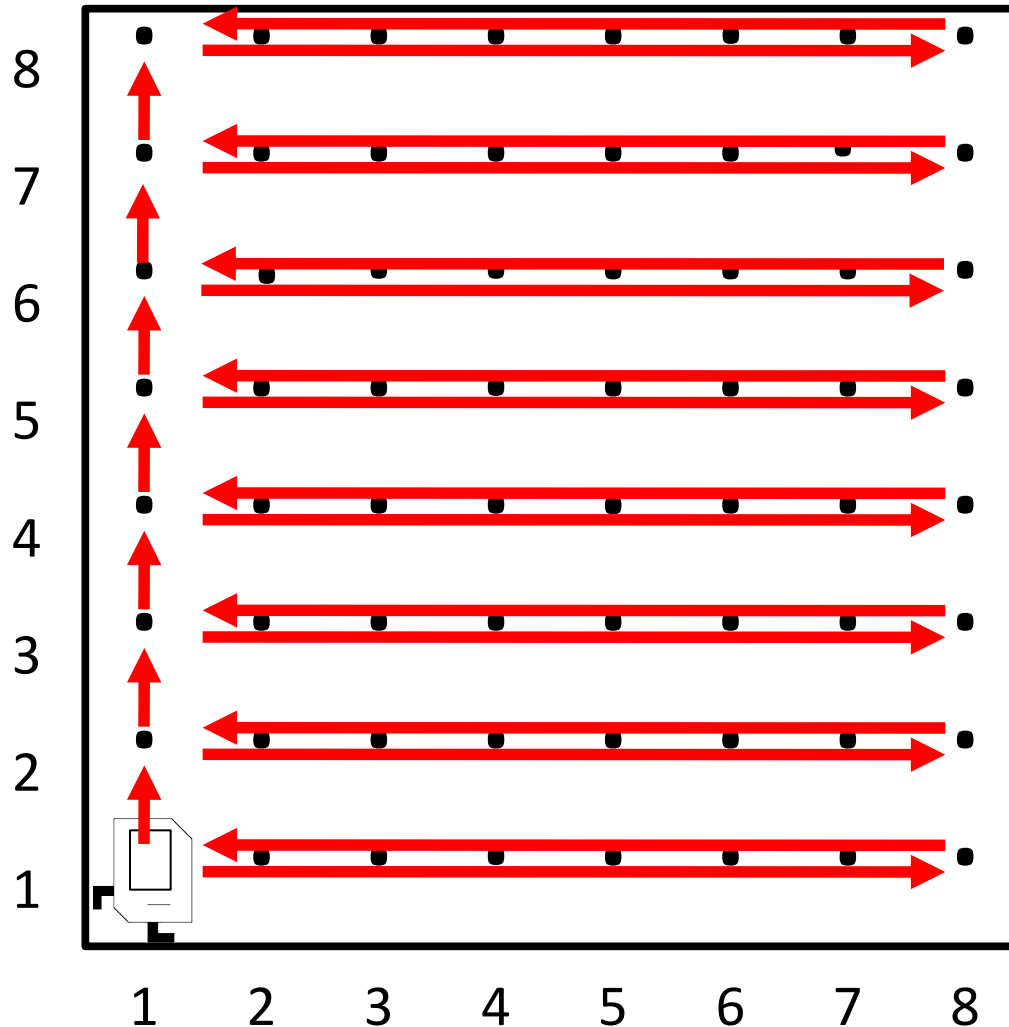
Possible Algorithm 2



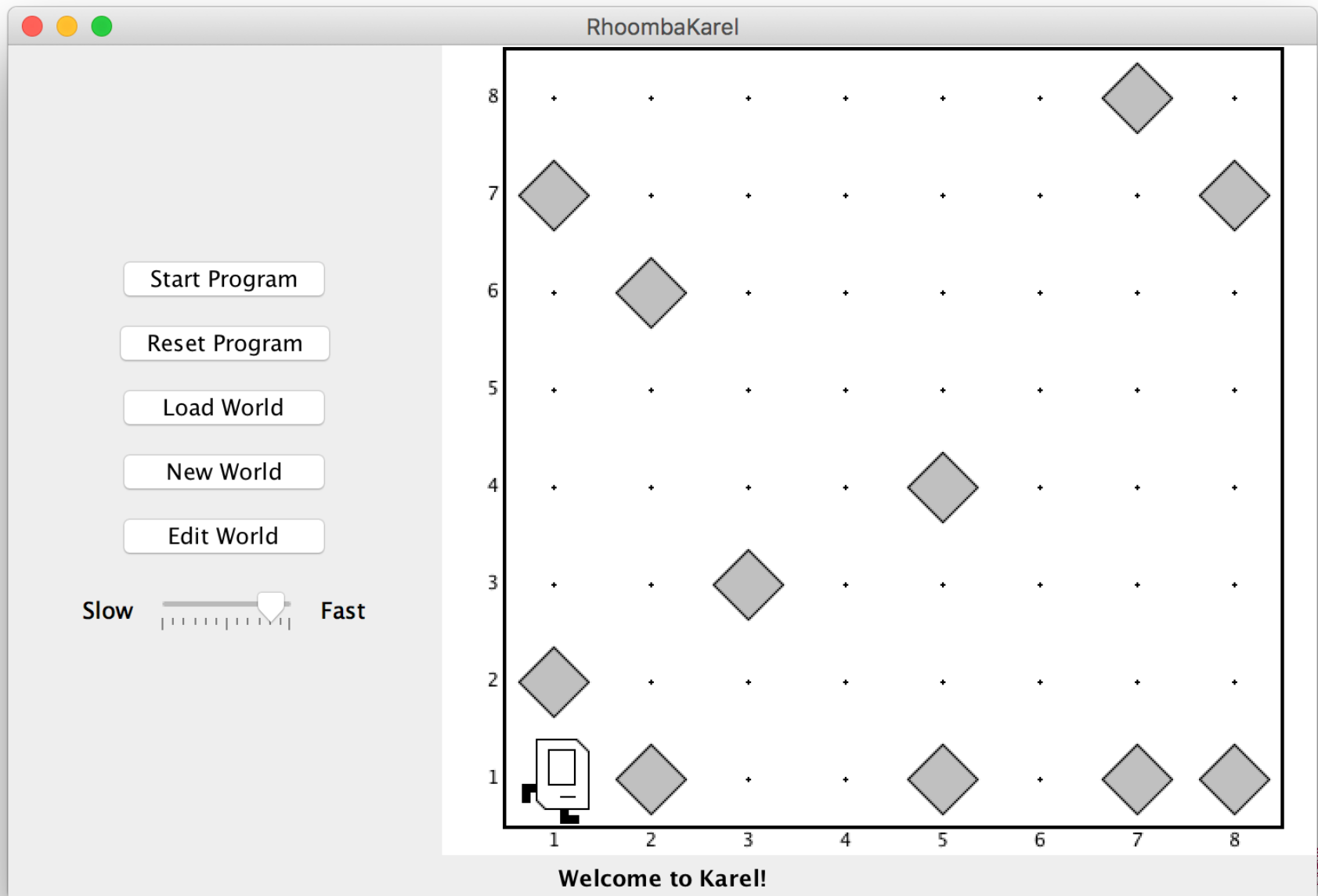
Possible Algorithm 3



Possible Algorithm 4



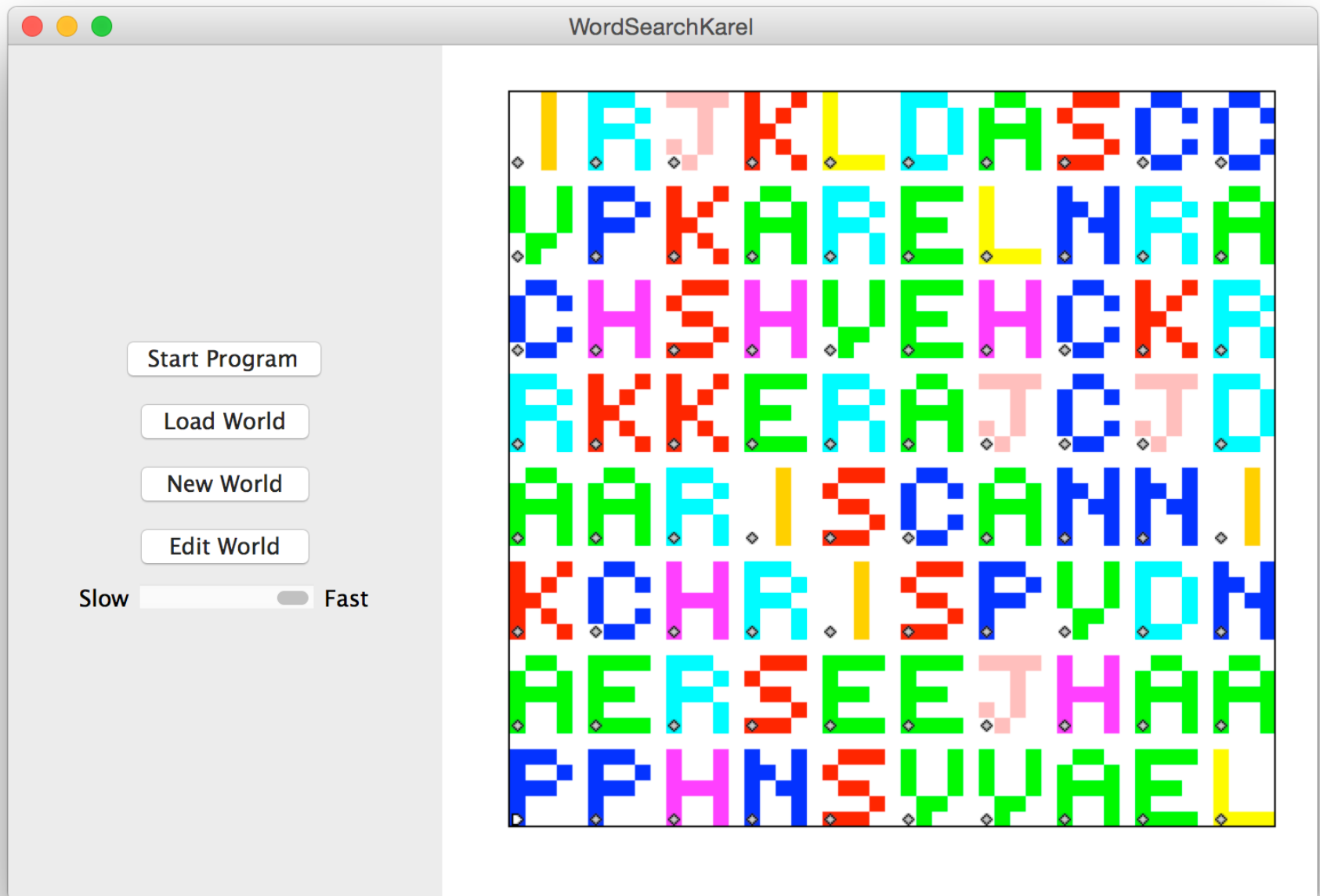
Rhoomba Karel



```
def friday(){  
    banishWinter()  
    decomposition()  
    doubleBeepers()  
    rhombaKarel()  
    if(extraTime()){  
        wordSearchKarel()  
    }
```

```
def friday(){  
    banishWinter()  
    decomposition()  
    doubleBeepers()  
    rhoombaKarel()  
    if(extraTime()){  
        wordSearchKarel()
```

```
def friday(){  
    banishWinter()  
    decomposition()  
    doubleBeepers()  
    rhombaKarel()  
    if(extraTime()){  
        wordSearchKarel()    }
```



Happy Friday