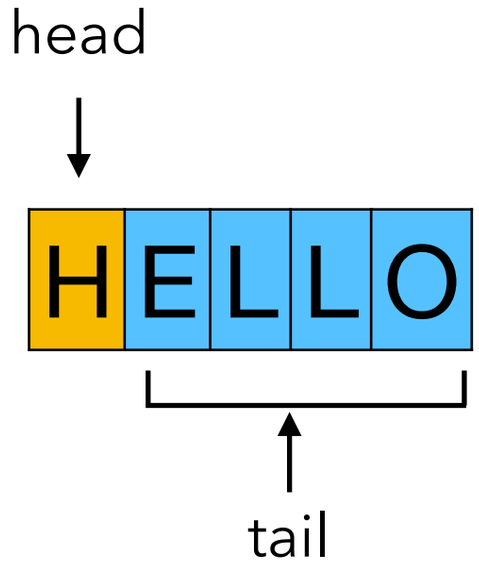
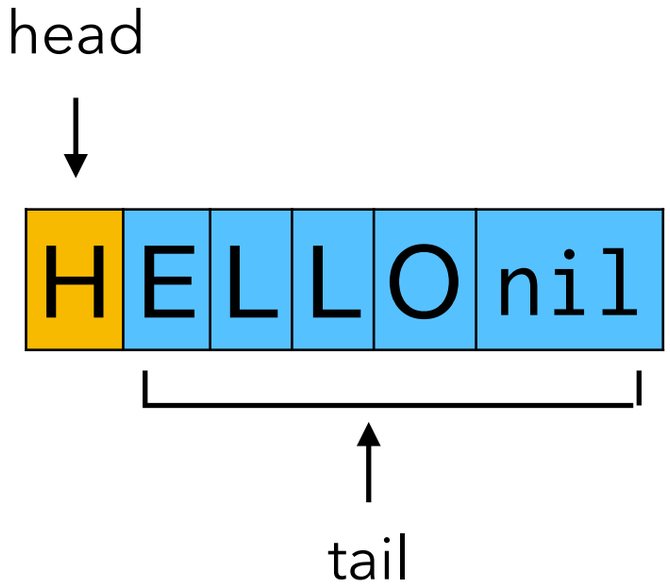
Learning to code

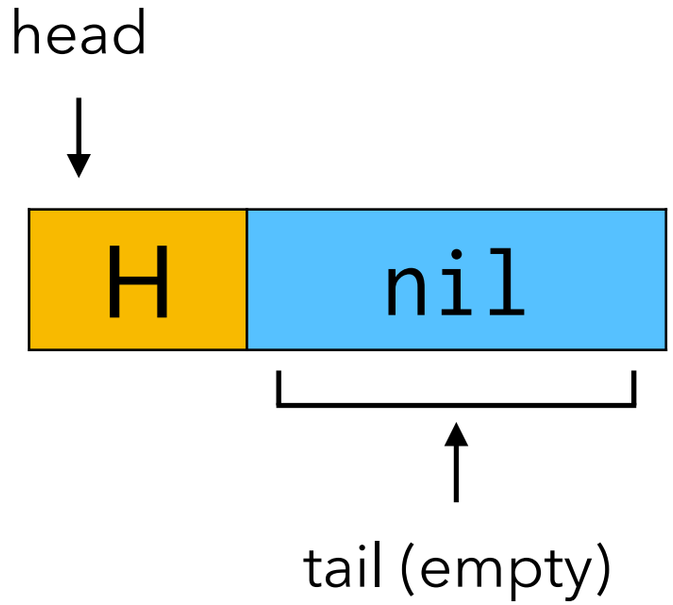
input	output
cat	c
dog	d
bear	**?**

input	output
cat	c
dog	d
bear	**b**

```
def f(a):
    return a[0]
```


head
HELLO
tail


head
H
E
L
L
O
nil
tail


head
H
nil
tail (empty)

```
def f(a):
    return a[0]
```

```
def f(a):
    return head(a)
```

input	output
cat	a
dog	o
bear	**?**

input	output
cat	a
dog	o
bear	**e**

```
def f(a):
    t = tail(a)
    return head(t)
```

input	output
dog	g
sheep	p
chicken	**?**

input	output
dog	g
sheep	p
chicken	**n**

```
def f(a):
    return a[-1]
```

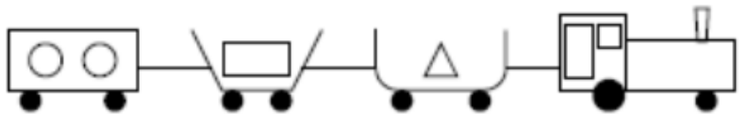
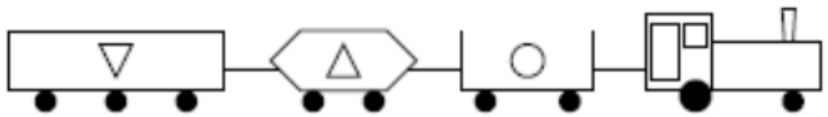
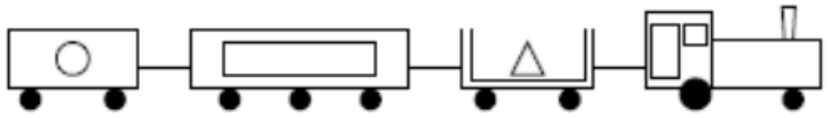
```
def f(a):
    t = tail(a)
    if empty(t):
        return head(a)
    return f(t)
```

input	output
ecv	cat
fqi	dog
iqqug	**?**

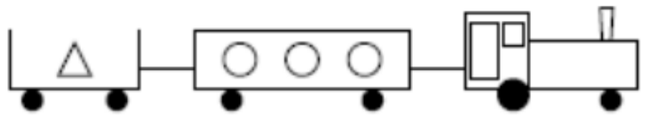
input	output
ecv	cat
fqi	dog
iqqug	**goose**

```
def f(a):
    if empty(a):
        return a
    b = head(a)
    c = ord(b)
    d = c-2
    e = chr(d)
    t = f(tail(a))
    return concat(e,t)
```

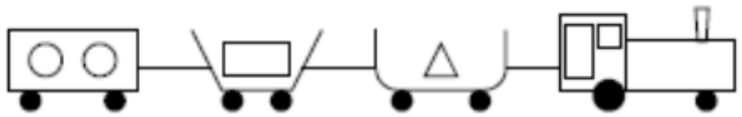
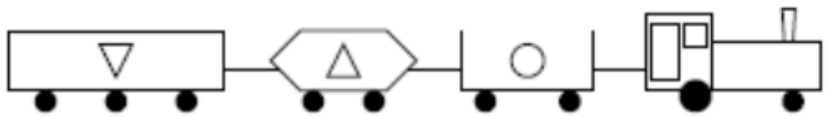
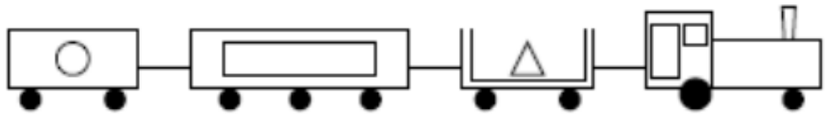
Good

1. 
2. 
3. 
4. 
5. 

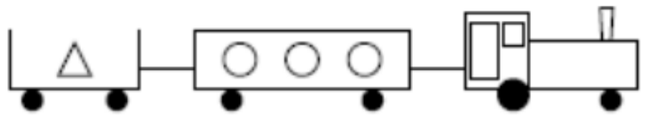
Bad

1. 
2. 
3. 
4. 
5. 

Good

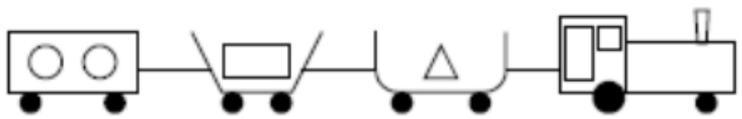
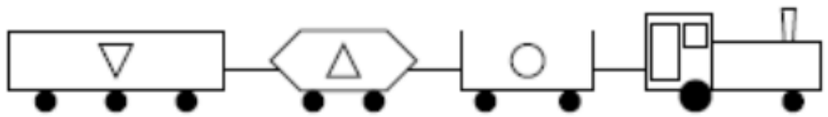
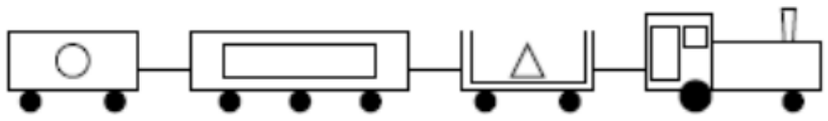
1. 
2. 
3. 
4. 
5. 

Bad

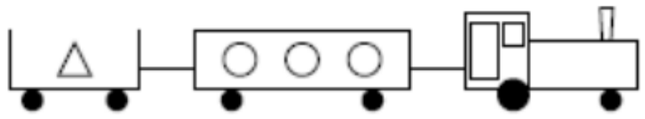
1. 
2. 
3. 
4. 
5. 

A train is good if it has a carriage that has **two wheels** and is **closed**

Good

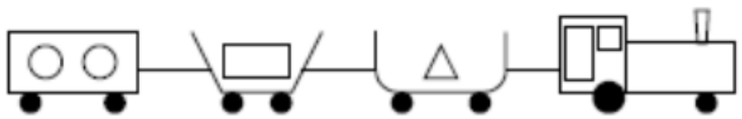
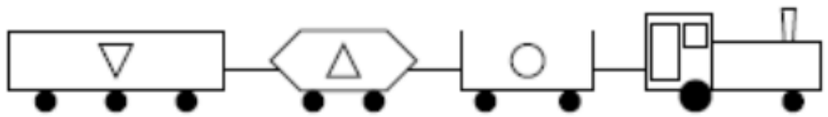
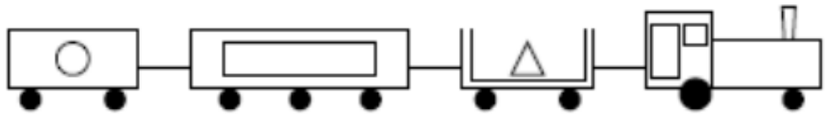
1. 
2. 
3. 
4. 
5. 

Bad

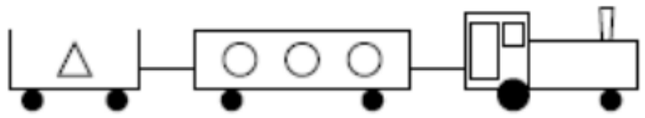
1. 
2. 
3. 
4. 
5. 

```
def good(train):
    for car in cars(train):
        if two_wheels(car) and closed(car):
            return True
    return False
```

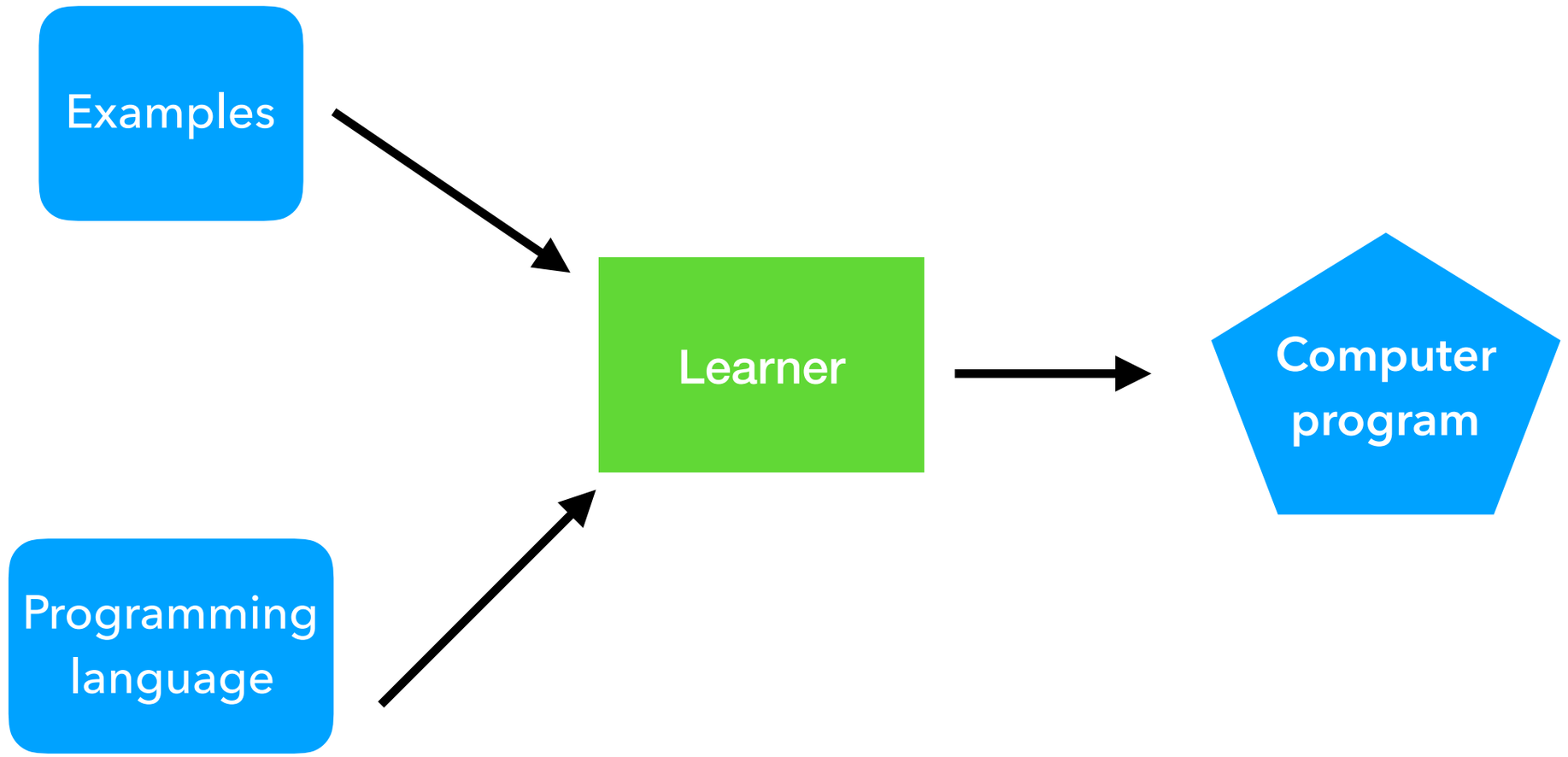
Good

1. 
2. 
3. 
4. 
5. 

Bad

1. 
2. 
3. 
4. 
5. 

```
good(A):-
    has_car(A,B),
    two_wheels(B),
    closed(B).
```


Examples
Learner
Computer program
Programming language

How?

- Set covering
- Divide-and-conquer
- Generate and test
- Proof search
- Neural networks

Applications

Repetitive tasks

	A	B	C
1	**Participants**	**Country**	
2	Ronnie Anderson, UK	UK	
3	Tom Boone, Canada	Canada	
4	Sally Brook, USA	USA	
5	Jeremy Hill, Australia	Australia	
6	Mattias Waldau, USA	USA	
7	Robert Furlan, France	France	
8	David White, UK	UK	

Learning game rules
(or solving programming puzzles)

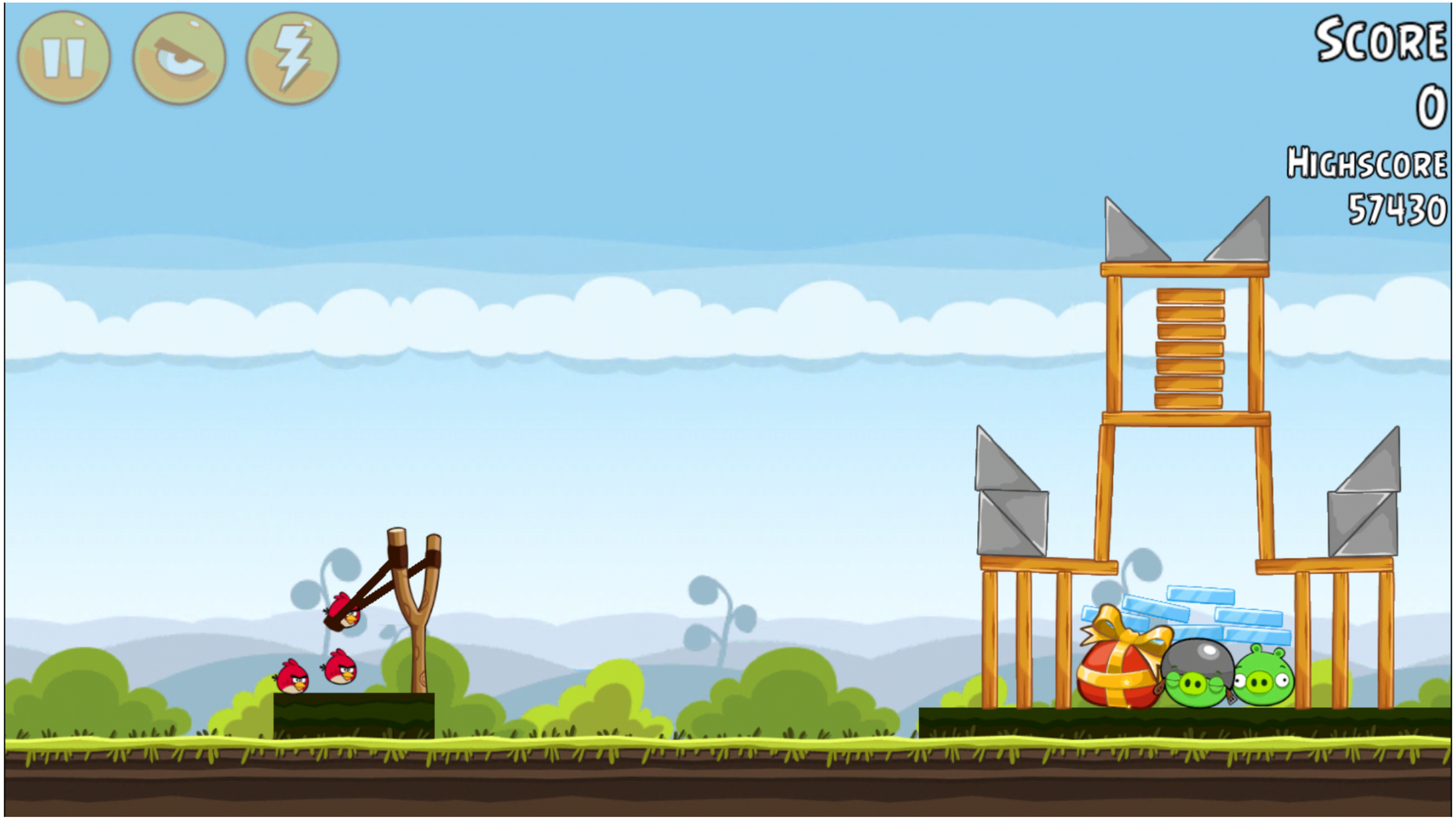
```
% examples
fizz(4) = 4
fizz(3) = fizz
fizz(10) = buzz
fizz(11) = 11
fizz(30) = fizzbuzz
```

```
def f(a):
    if div3(a) and not div5(a):
        return 'fizz'
```

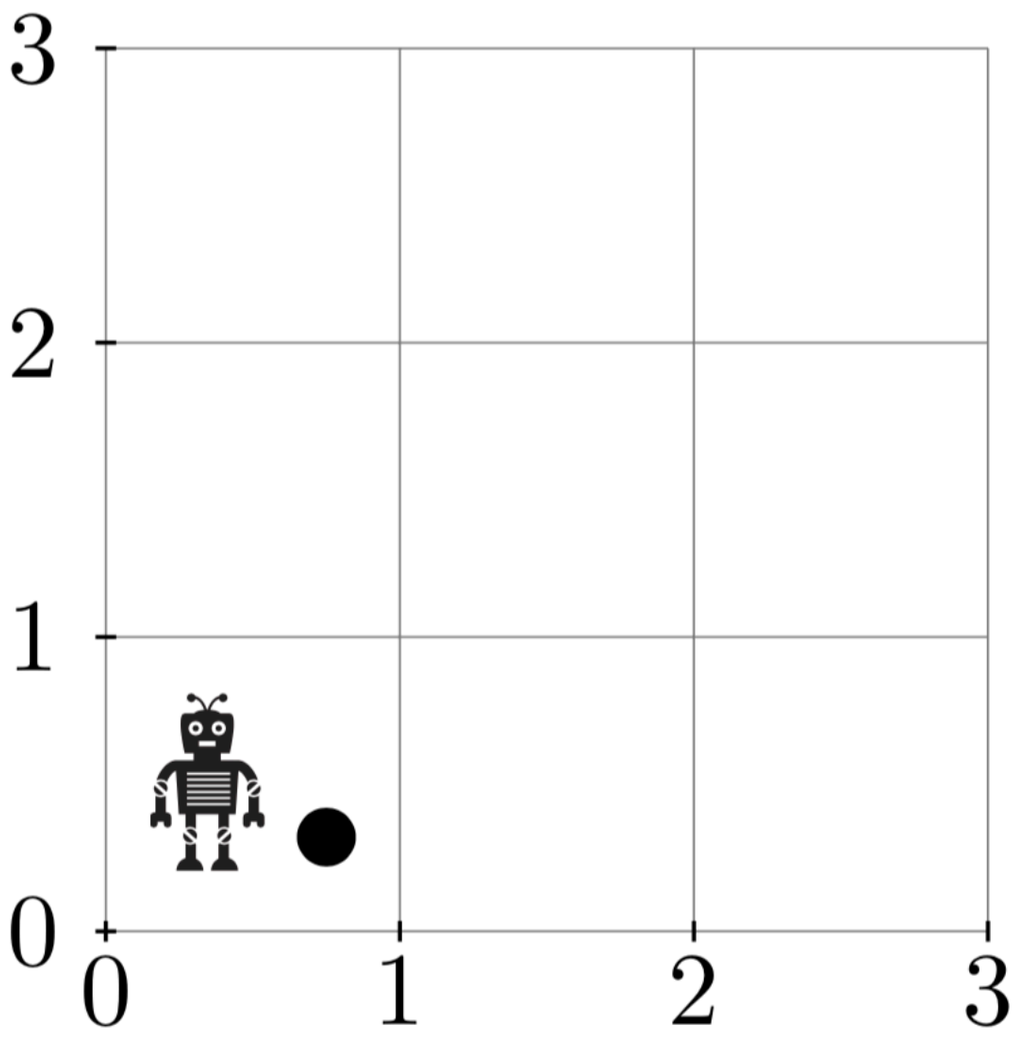
```
def f(a):
    if div3(a) and not div5(a):
        return 'fizz'
    if not div3(a) and div5(a):
        return ‘buzz'
```

```
def f(a):
    if div3(a) and not div5(a):
        return 'fizz'
    if not div3(a) and div5(a):
        return 'buzz'
if div3(a) and div5(a):
    return ‘fizzbuzz'
```

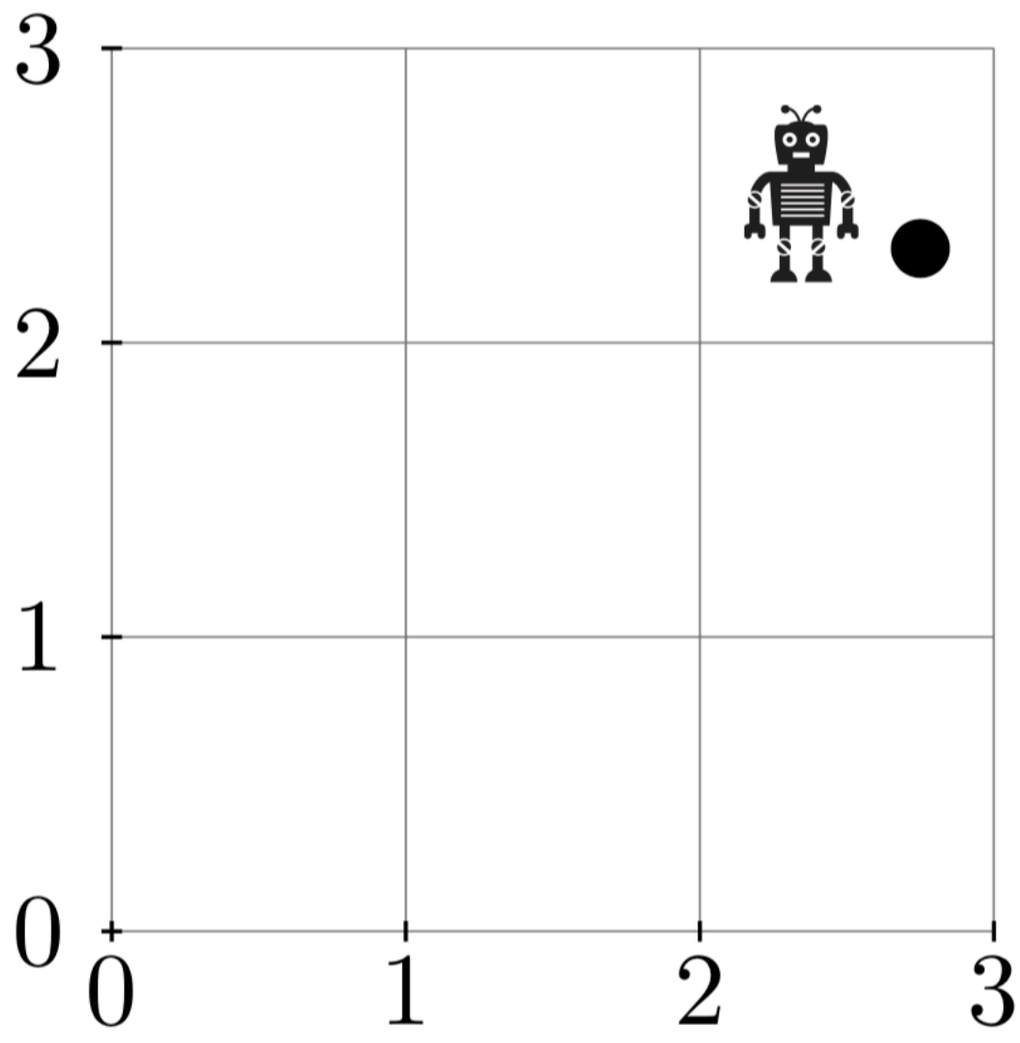
```
def f(a):
    if div3(a) and not div5(a):
        return 'fizz'
    if not div3(a) and div5(a):
        return 'buzz'
if div3(a) and div5(a):
    return 'fizzbuzz'
return a
```


SCORE
0
HIGHSCORE
57430

Learning robot strategies



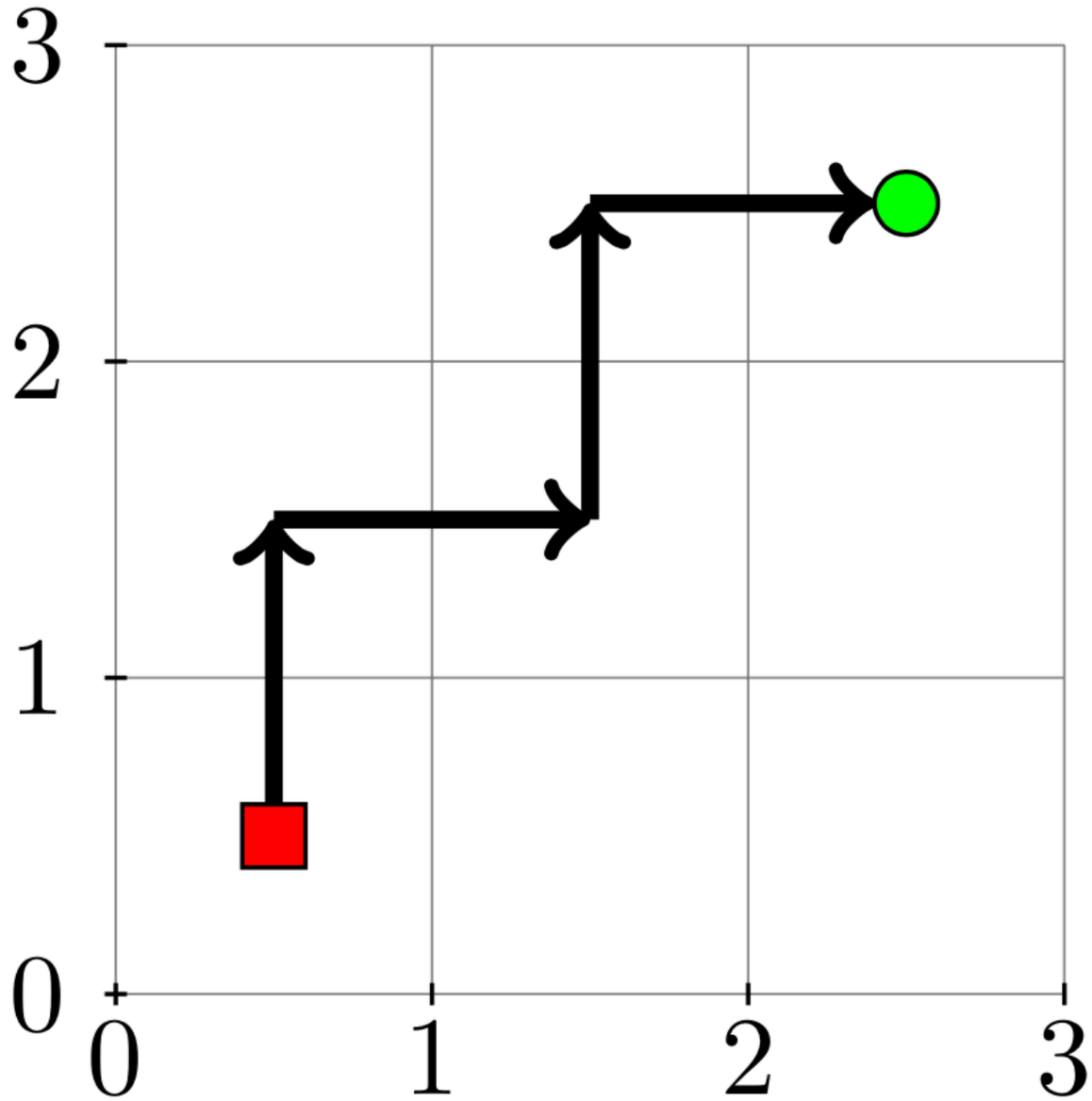
(a) Initial state



(b) Final state

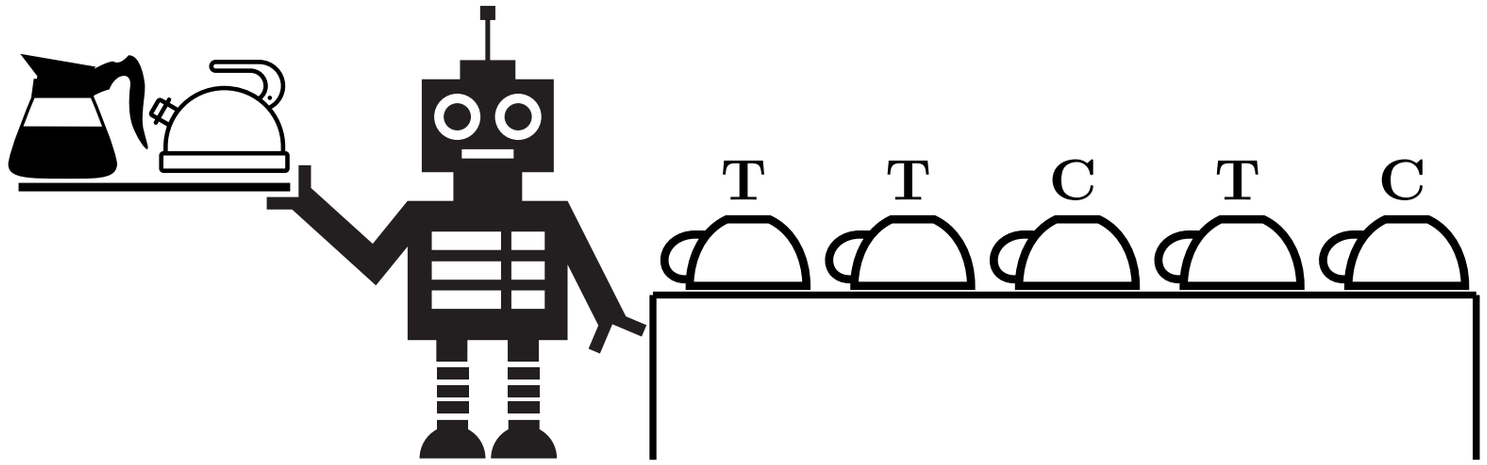
```
f(A,B):-
      grab(A,C),
      f1(C,D),
      f1(D,E),
      drop(E,B).
f1(A,B):-
      up(A,C),
      right(C,B).
```

(a) Program



(b) Plan

Initial state:



Final state:

Lifelong learning

task	input	output
f	philip.larkin@sj.ox.ac.uk	Philip Larkin

task	input	output
f	philip.larkin@sj.ox.ac.uk	Philip Larkin

```
def f(a):
    b = uppercase(a)
    c = copyword(b)
    d = skip1(c)
    e = space(d)
    f = uppercase(e)
    g = copyword(f)
return skiprest(g)
```

10 seconds

task	input	output
g	tony	Tony

task	input	output
g	tony	Tony

```
def g(a):
    b = uppercase(a)
    return copyword(b)
```

task	input	output
g	tony	Tony
f	philip.larkin@sj.ox.ac.uk	Philip Larkin

```
def g(a):
    b = uppercase(a)
    return copyword(b)
```

task	input	output
g	tony	Tony
f	philip.larkin@sj.ox.ac.uk	Philip Larkin

```
def g(a):
    b = uppercase(a)
    return copyword(b)

def f(a):
    b = g(a)
    c = skip1(b)
    d = space(c)
    e = g(b)
    return skiprest(e)
```

task	input	output
g	tony	Tony
f	philip.larkin@sj.ox.ac.uk	Philip Larkin

```
def g(a):
    b = uppercase(a)
    return copyword(b)

def f(a):
    b = g(a)
    c = skip1(b)
    d = space(c)
    e = g(b)
    return skiprest(e)
```

2 seconds!

Learning efficient programs

input	output
[s,h,e,e,p]	e
[a,l,p,a,c,a]	a
[c,h,i,c,k,e,n]	**?**

input	output
[s,h,e,e,p]	e
[a,l,p,a,c,a]	a
[c,h,i,c,k,e,n]	**c**

```
def f(a):
    h = head(a)
    t = tail(a)
    if h in t:
        return h
    return f(t)
```

```
def f(a):
    return g(sorted(a))

def g(a):
    h = head(a)
    t = tail(a)
    if h = head(t)
        return h
    return g(t)
```

- Automation
  - Better software
- Scientific discovery
  - New algorithms
  - Other sciences