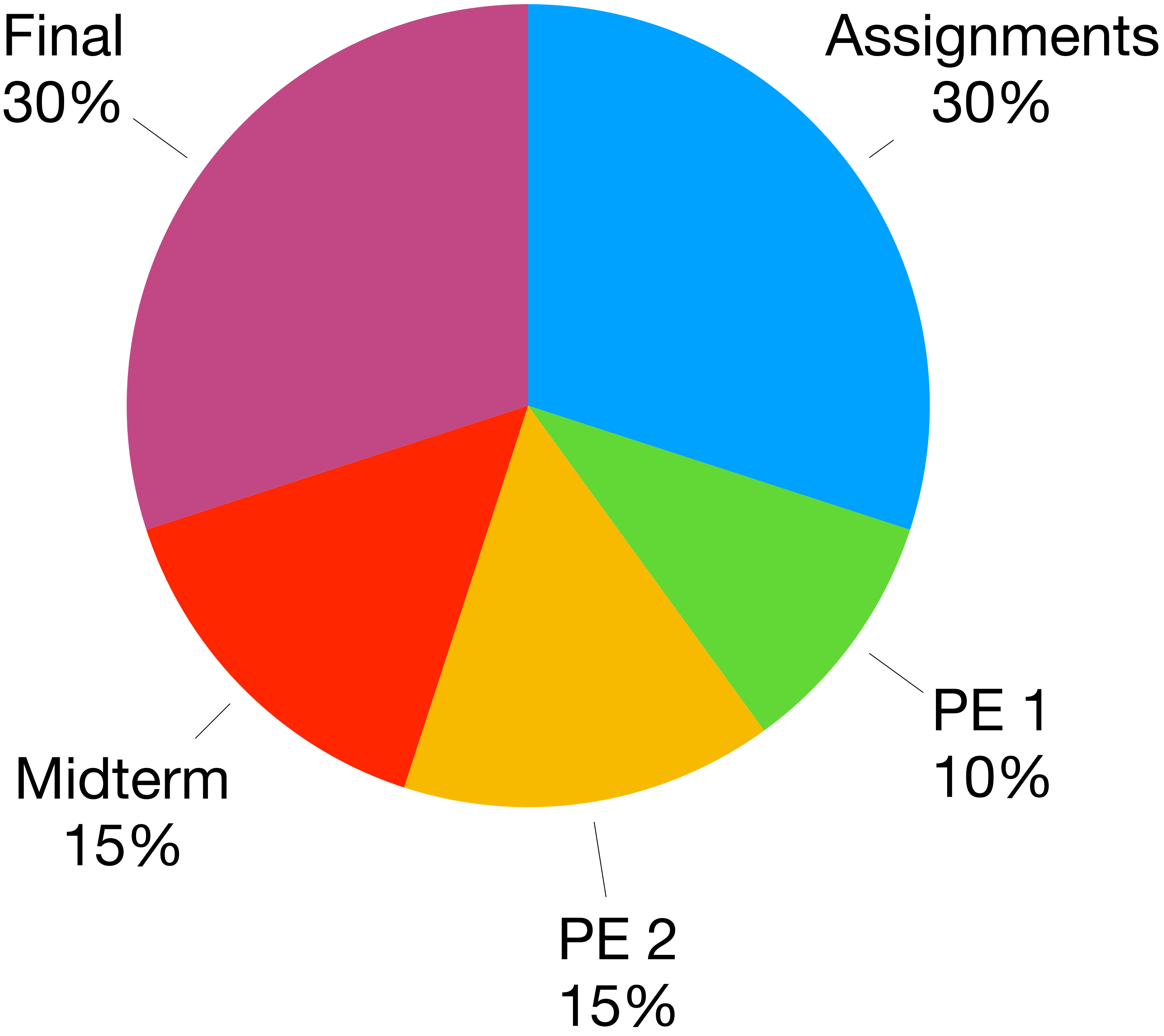
Lecture 12

13 November 2018

Admin Matters
Recap and Look Ahead

Final Assessment

27 November 2018
Morning


Final
30%
Assignments
30%
PE 1
10%
PE 2
15%
Midterm
15%

Scope

Everything from
Unit 1 to Unit 28

Format

Some MCQs
Some Short Questions

Open Book

Nothing to Memorize

Focus on Understanding
and Applying Principles

Assignment 9

university policy: no deadlines during reading week no change in deadline

but no late penalty until after
18 November Sunday 23:59

Deadline to Finalize Marks

5 December 2018
6:00pm

Looking Forward

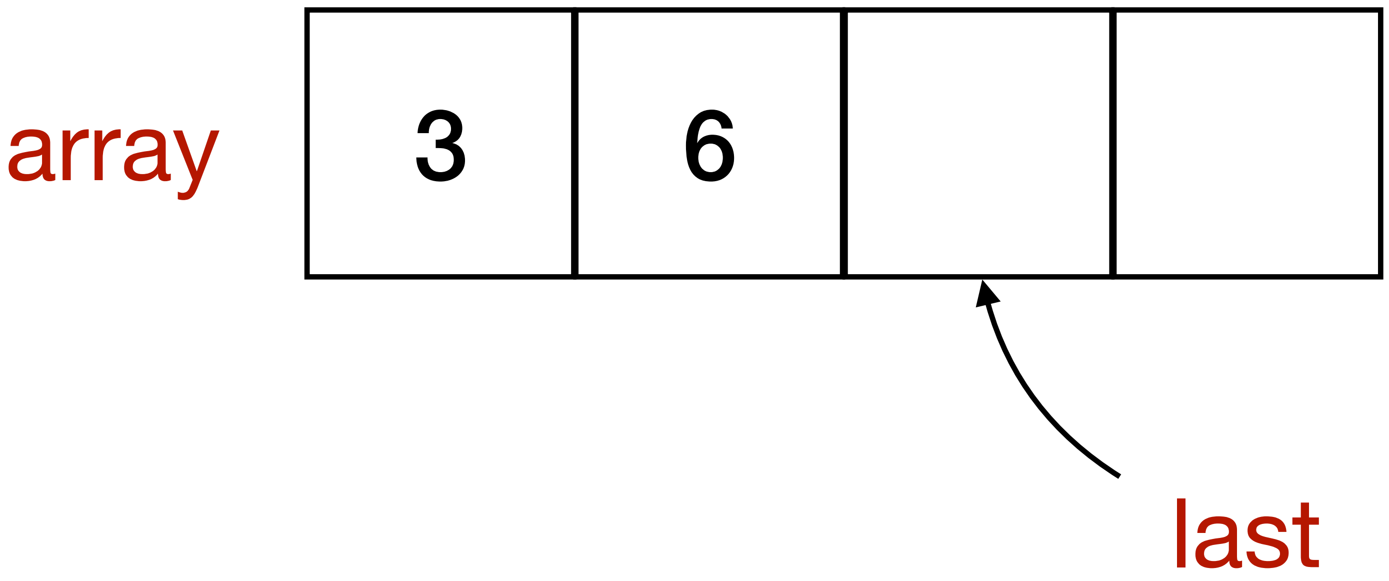
Teaser to CS2040C

Data Structures

- A collection of data values and the operations that can be applied
- How to organise and manage data for efficient access and modification?

Example: List

- An ordered list of numbers
- Can
  - create a list
  - append to the list
  - remove a number from the list
  - find the position of a number
  - destroy a list
  - print a list



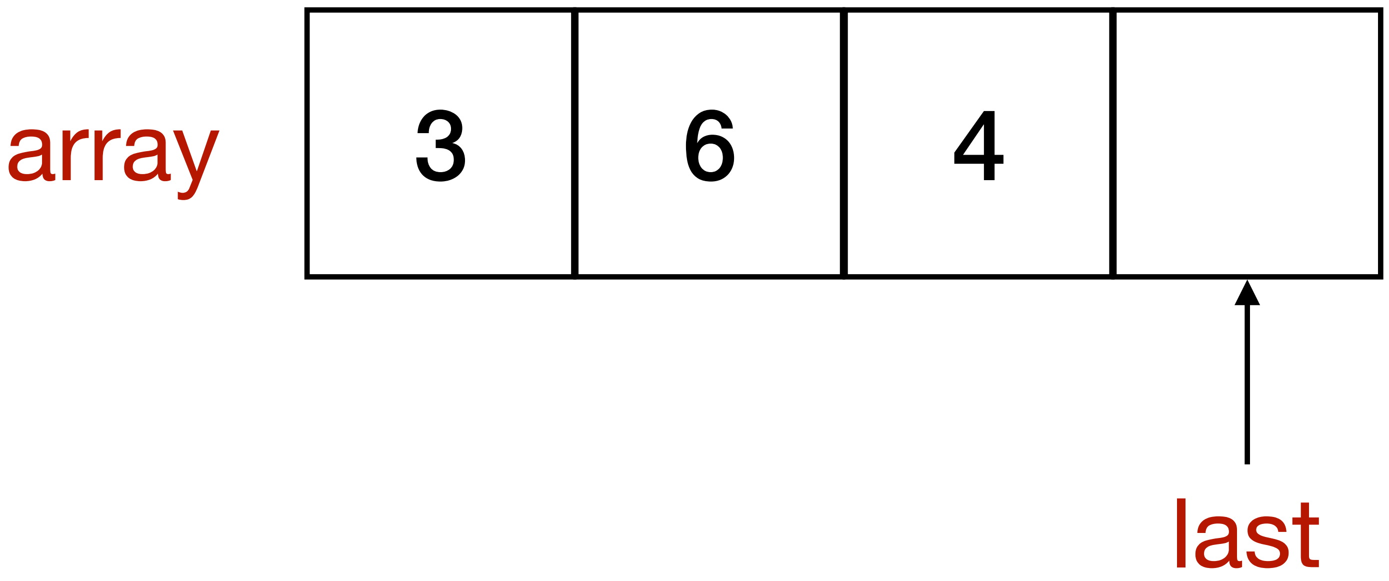
array

3	6		

last

size = 4

append 4



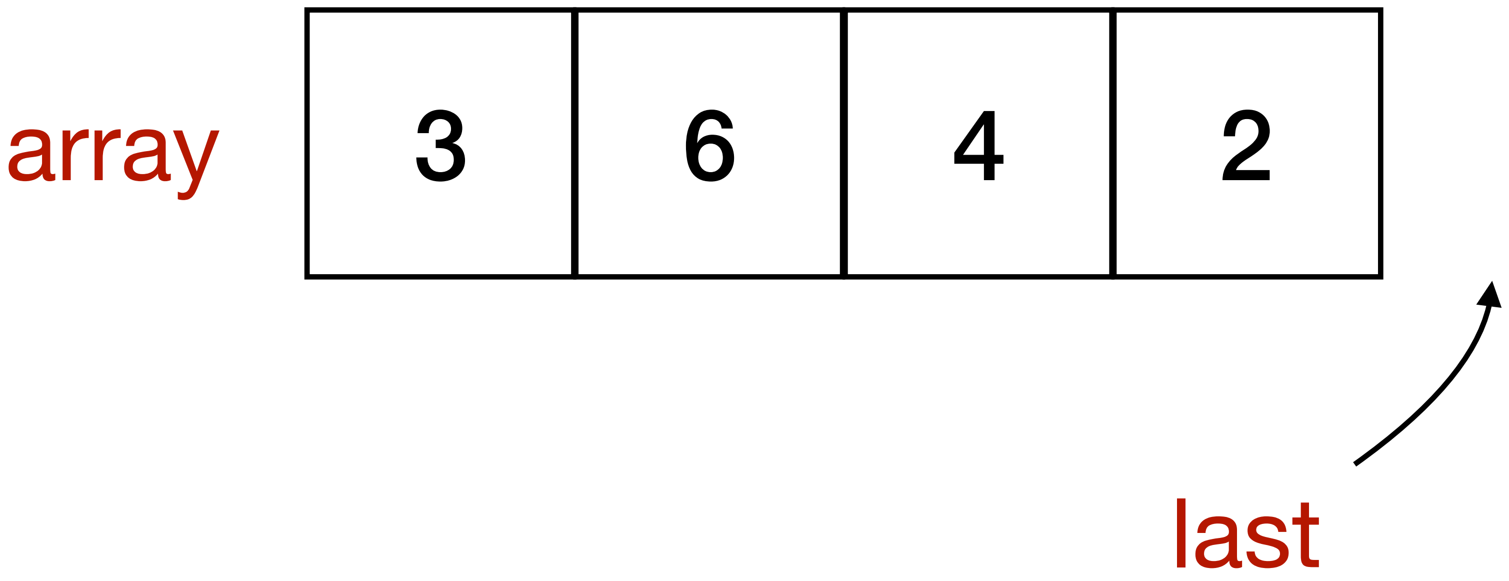
array

3 6 4

last

size = 4

append 2



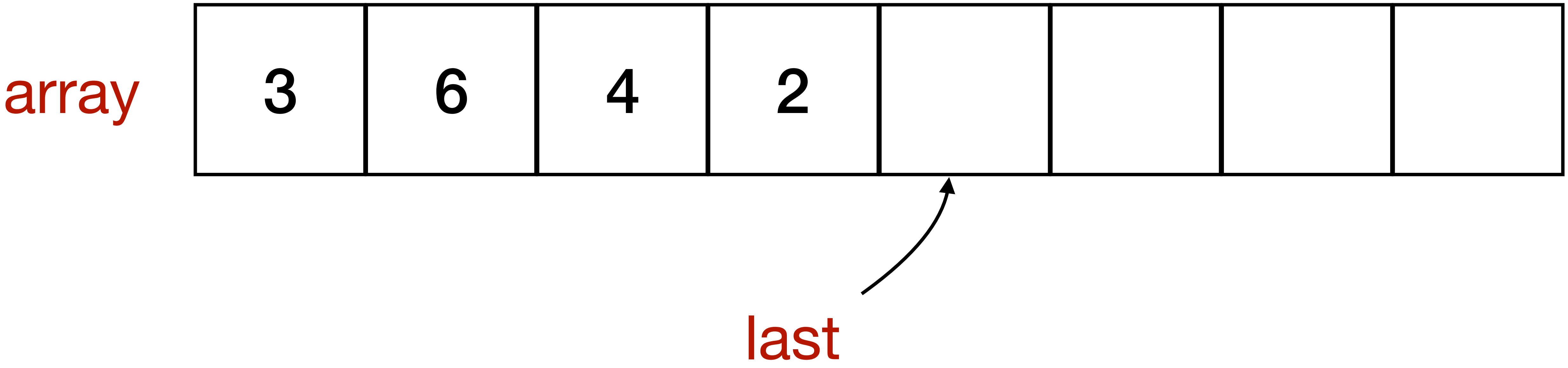
array

3	6	4	2

last

size = 4

append 2



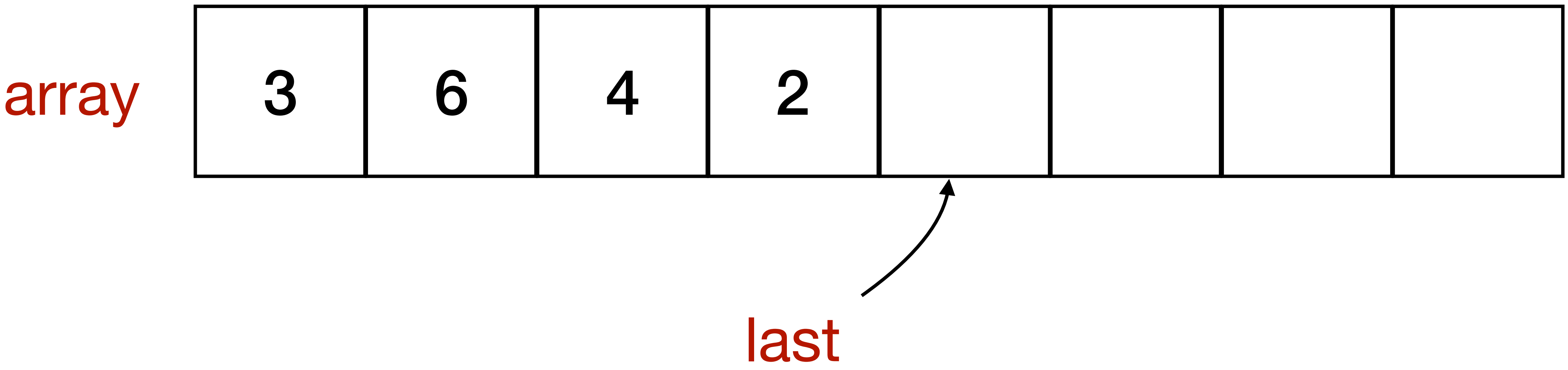
array

3	6	4	2				

last

size = 8

delete 6



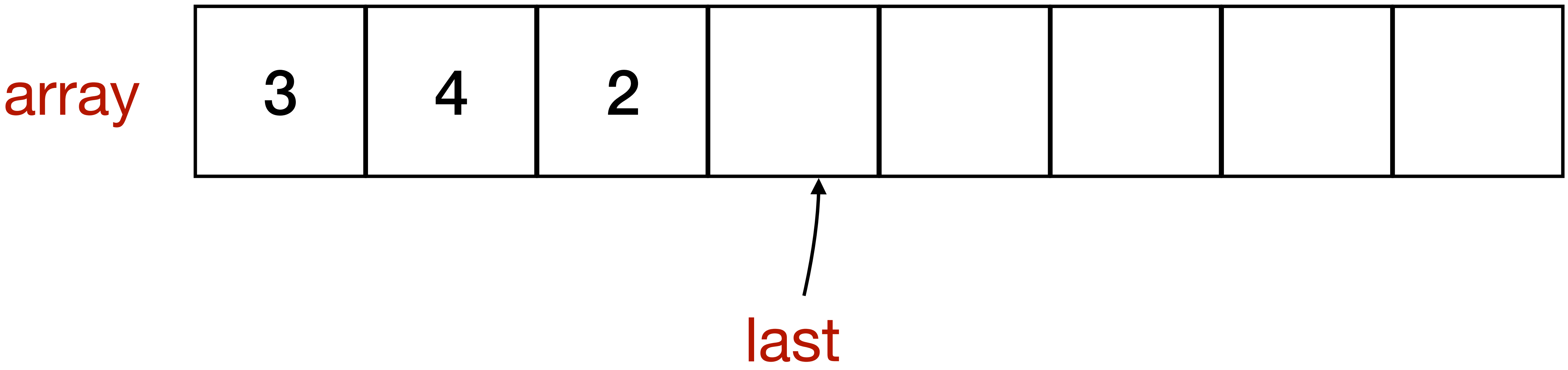
array

3	6	4	2				

last

size = 8

delete 6



array

3	4	2					

last

size = 8

What will you learn?

How to write C

How a C program behaves

Tools and techniques to help write good and correct C programs

How to use C to solve computational problems

Learning to write a program that does what you want it to do is actually not difficult.

Knowing what you want your program to do is the more challenging part!

Tools and techniques to help write good and correct C programs

Tools / Good Practice

- clang
- vim
- bash
- lldb*
- make*
- git*
- clang-tidy*

- assertion
- good programming style
- (some) secure programming
- writing documentation
- testing

How to write C

C Language / Syntax

- Types
- Functions
- + - * / %
- if else
- && || !
- for / while / do-while

- arrays
- pointers & *
- calloc / free
- #include / #define
- struct
- printf / scanf

Things We Didn’t Cover

- FILE I/O
- argc / argv
- enum
- string functions
- bit operations
- separate compilation

How a C program behaves

Behavourial / Mental Model

- machine code
- data in memory
- types
- stack and heap
- call stack
- memory address
- call by value / reference

How to use C to solve computational problems

Problem Solving

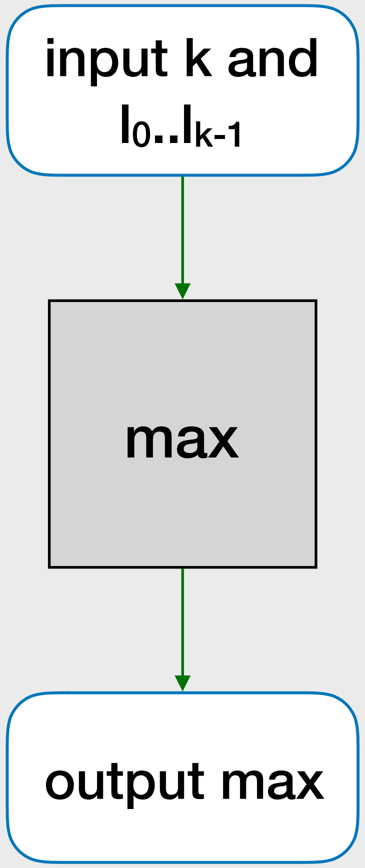
- Decomposition
- Recursion & Backtracking
- Flowcharts
- Conditionals
- Loop
- Assertion and Invariants
- Arrays and Lists
- Sorting
- Searching
- Efficiency

Computational Thinking

The mental process associated with computational problem solving

1. Decomposition
2. Pattern Recognition
3. Abstraction
4. Algorithms

Decomposition

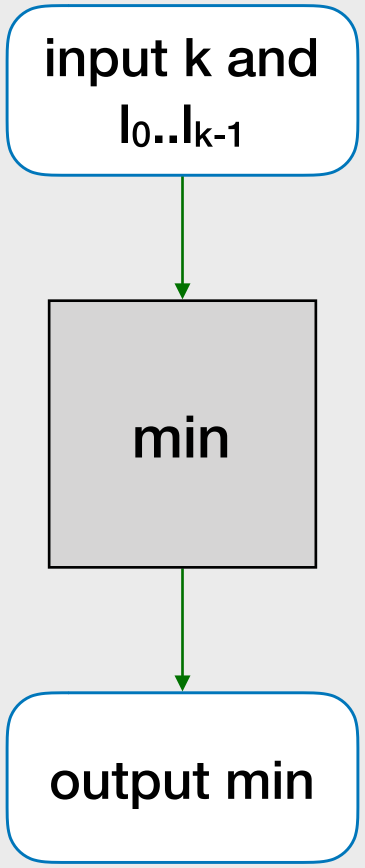


input k and $l_0..l_{k-1}$

max

output max

max(L, k)

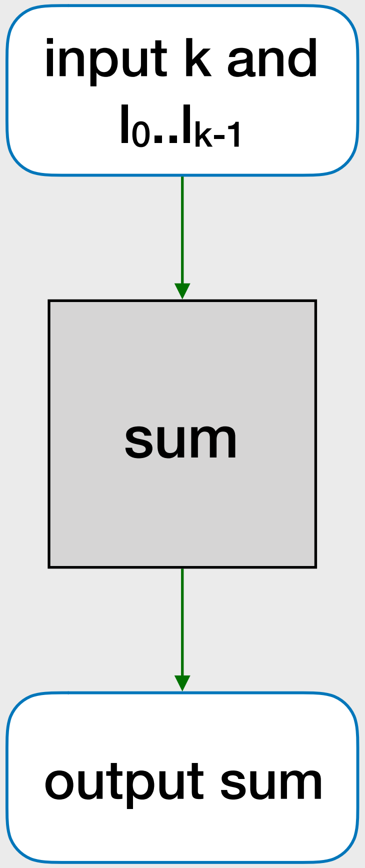


input k and $l_0..l_{k-1}$

min

output min

min(L, k)



input k and $l_0..l_{k-1}$

sum

output sum

sum(L, k)

Find the Std Dev

- Give an algorithm to find the standard deviation of a given list L of k integers.

$$\sqrt{\frac{\sum_{i=0}^{k-1}(l_i - \mu)^2}{k}}$$

Break it down to subproblems

$$\sqrt{\frac{\sum_{i=0}^{k-1}(l_i - \mu)^2}{k}}$$

sqrt
mean
square
subtract

taxi

```
double surcharge(long day, long hour, long minute)
{
  if (is_weekday(day) && is_morning_peak(hour, minute)) {
    return MORNING_SURCHARGE;
  }
  if (is_evening_peak(hour)) {
    return EVENING_SURCHARGE;
  }
  if (is_midnight_peak(hour)) {
    return MIDNIGHT_SURCHARGE;
  }
  return 1.0;
}
```

```
bool is_weekday(long day)
{
  return (day >= 1 && day <= 5);
}

bool is_morning_peak(long hour, long minute)
{
  return (hour >= 6 && hour < 9) || (hour == 9 && minute <= 29)
}

bool is_evening_peak(long hour)
{
  return (hour >= 18 && hour <= 23);
}

bool is_midnight_peak(long hour)
{
  return (hour >= 0 && hour < 6);
}
```

social

```c
bool is_friend(char **network, int i, int j) {
  if (i >= j) {
    return network[i][j] == FRIEND;
  }
  return network[j][i] == FRIEND;
}
```

```
/**
 * Checks if i and j has a common friend
 * @param[in] n The number of users
 * @param[in] degree_1 The 1-hop friendship information
 * @param[in] degree_h The h-hop friendship information
 * @param[in] i A user
 * @param[in] j Another user
 * @return FRIEND if i and j has a (h+1)-hop connection,
 *         STRANGER otherwise.
 */
char is_connected(long n, char **degree_1, char **degree_h,
    long i, long j) {
  for (int m = 0; m < n; m += 1) {
    if (is_friend(degree_1, i, m) &&
        is_friend(degree_h, m, j)) {
      return FRIEND;
    }
  }
  return STRANGER;
}
```

```c
/**
 * Computers the h-hop friendship for the whole network.
 * @param[in] n           The number of users.
 * @param[in] degree_1    The 1-hop friendship network
 * @param[in] degree_h    The h-hop friendship network
 * @param[out] degree_h1  The (h+1)-hop friendship network
 */
void compute_degree_h(long n, char **degree_1, char **degree_h,
    char **degree_h1) {
  for (int i = 0; i < n; i += 1) {
    for (int j = 0; j <= i; j += 1) {
      if (is_friend(degree_h, i, j)) {
        degree_h1[i][j] = FRIEND;
      } else {
        degree_h1[i][j] = is_connected(n, degree_1, degree_h,
            i, j);
      }
    }
  }
}
```

Decomposition

break complex problems down
into “bite-size” subproblems
that you can solve

George Pólya said

“If you can’t solve a problem,
then there is an easier problem
you can solve: find it”

Solve easier problem, then generalised

e.g.,

find two hops neighbors, then generalise to k hops

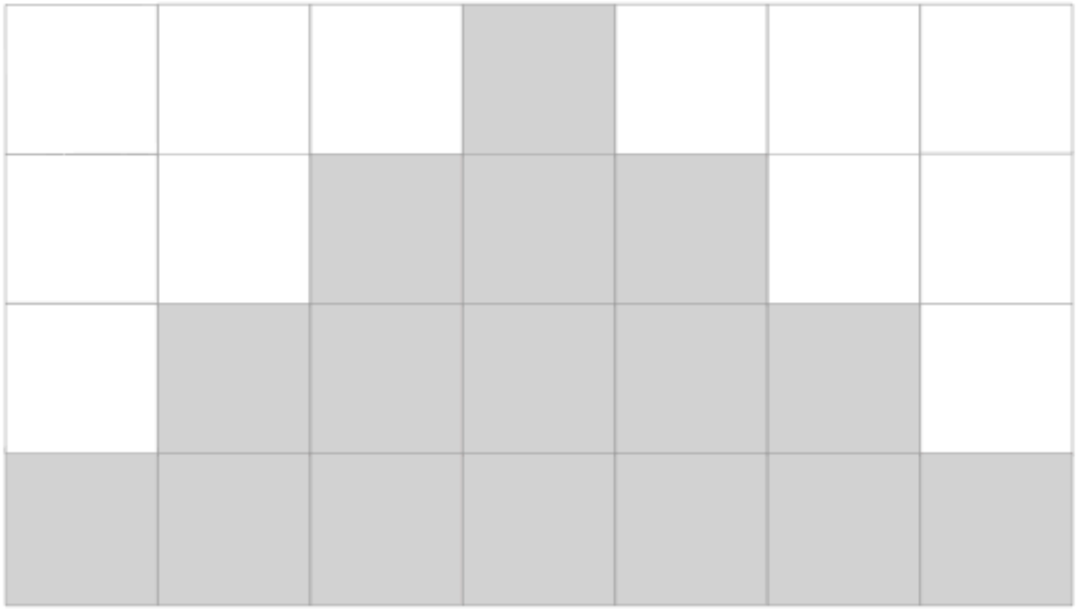
```
                            #
                           # #
                          ## ##
                         # # # #
                        #### ####
                       #   # #   #
                      ###### ######
                     # # # # # # # #
                    ## ## ## ## ## ##
                   # #   # # # #   # #
                  ########## ##########
                 #   # #   # #   # #   #
                ############ ############
               # # #   # # # # # #   # # #
              ## #  ##  # ## ## #  ##  # ##
             # # # # # # # # # # # # # # # #
            ################ ################
           #   # #   # #   # #   # #   # #   #
          ################## ##################
         # #   # # # #   # # # #   # # # #   # #
        ## ##  # ## #  ## ## ## ##  # ## #  ## ##
       # # # # #   # # # # # # # # # #   # # # # #
      ###################### ######################
     #   # #   # #   # #   # #   # #   # #   # #   #
    #### #### #### #### #### #### #### #### #### ####
   # # # # # #   # # # # # # # # # # # #   # # # # # #
  ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ##
 # # #   # # # # # #   # # # # # #   # # # # # #   # # #
############################ ############################
```

e.g.,
draw left
most cells,
then draw
the rest.

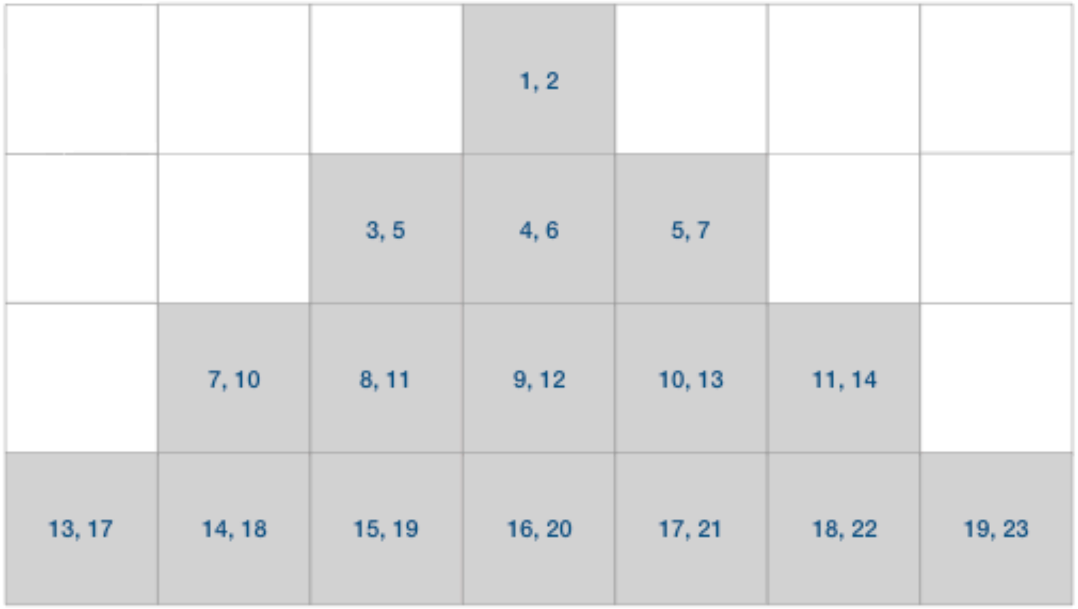
Pattern Recognition

pattern

See the figures below:



The figure above shows the shape of the triangle with height 4. The shaded locations belong to the triangle. Each square represents a cell.



vote

```
double percentage_of_nixon(long nixon, long total) {
    return nixon*100.0/total;
}

double percentage_of_mcneal(long mcneal, long total) {
    return mcneal*100.0/total;
}
```

```
double percentage(long nvotes, long total) {
    return nvotes*100.0/total;
}
```

taxi

Question 4: Taxi Fare (15 marks)

The taxi fare structure in Singapore must be one of the most complex in the world! Check out: http://www.taxisingapore.com/taxi-fare/.

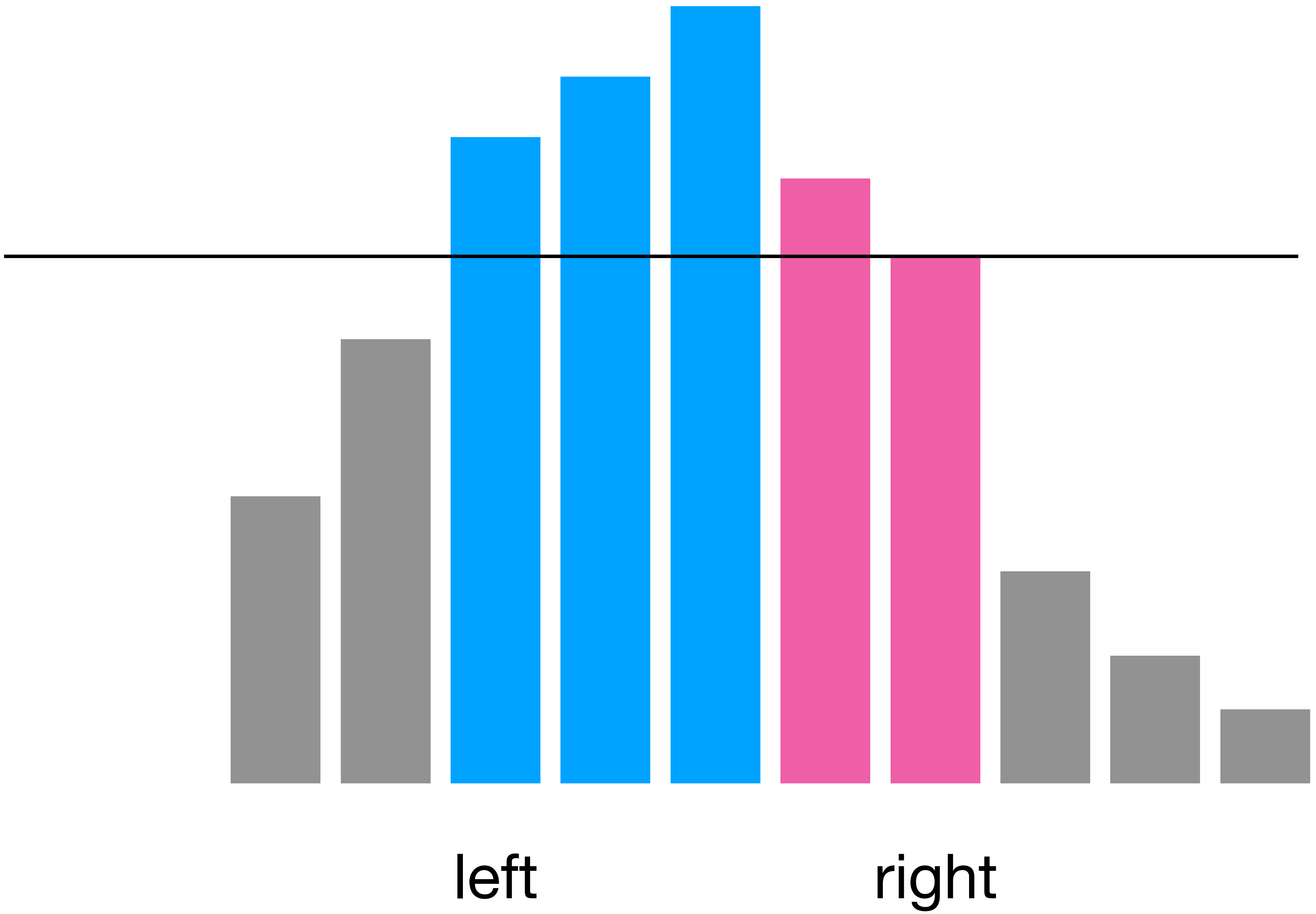
For the purpose of this exercise, we will just use the following simplified fare structure:

Basic Fare	
The first 1 km or less (Flag Down)	$3.40
Every 400 m thereafter or less, up to 10.2 km	$0.22
Every 350 m thereafter or less, after 10.2 km	$0.22

unit_distance	max_distance	fare
Every 1000 m	next 1 km	$3.40
Every 400 m	next 9.2 km	$0.22
Every 350 m	next ∞ km	$0.22

```
double fare = 0;
for (int i = 0; i < NUM_TIERS; i += 1) {
    if (distance < 0) {
      return fare;
    }
    long min_dist = min(distance, tiers[i].max_distance);
    fare += tiers[i].fare * ceil(min_dist*1.0 / tiers[i].unit_distance);
    distance -= tiers[i].max_distance;
 }
```

inversions


left
right

Pattern Recognition

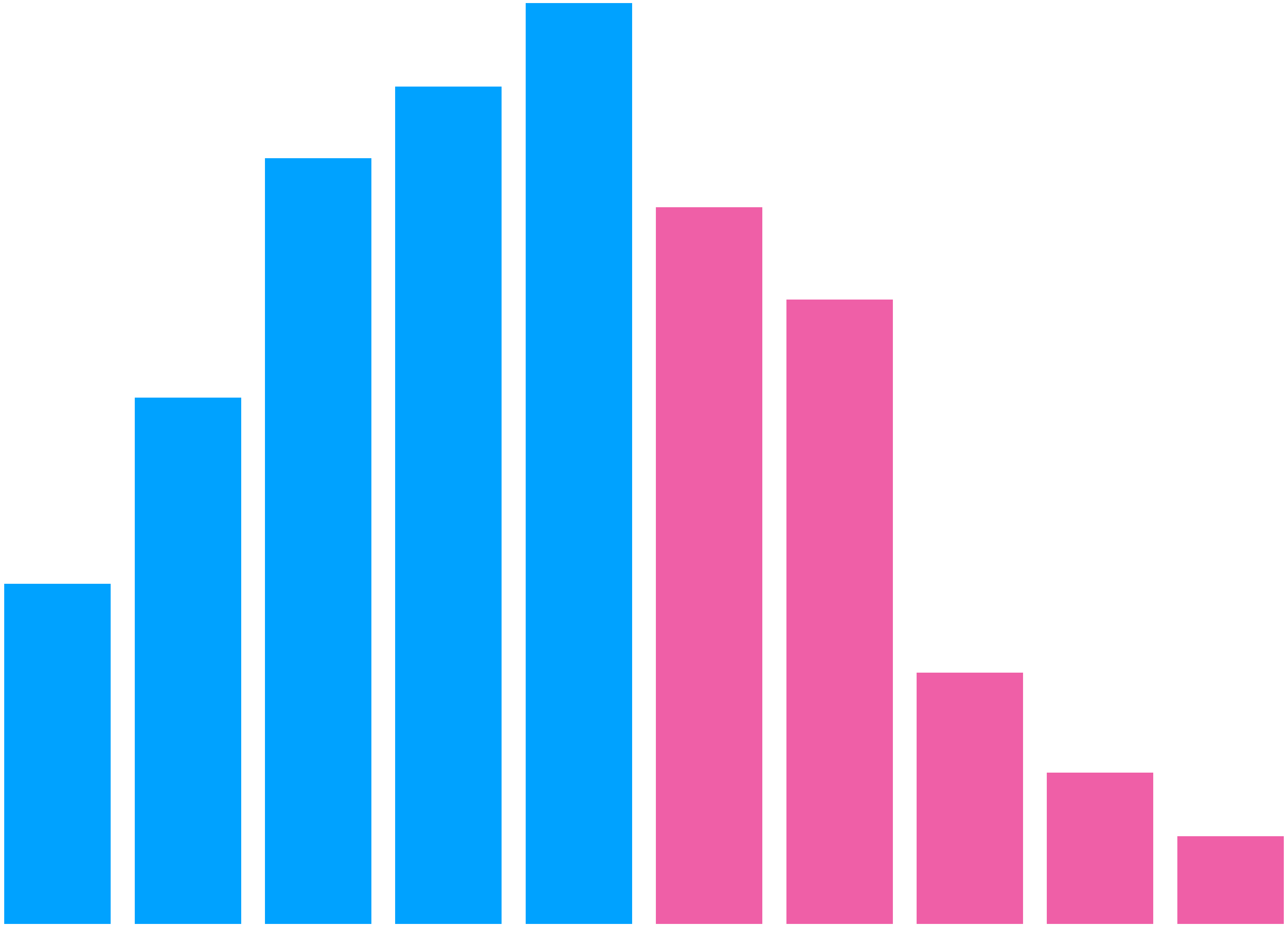
observe trends and patterns,
then generalise

Abstraction

Question 1: Peak (10 marks)

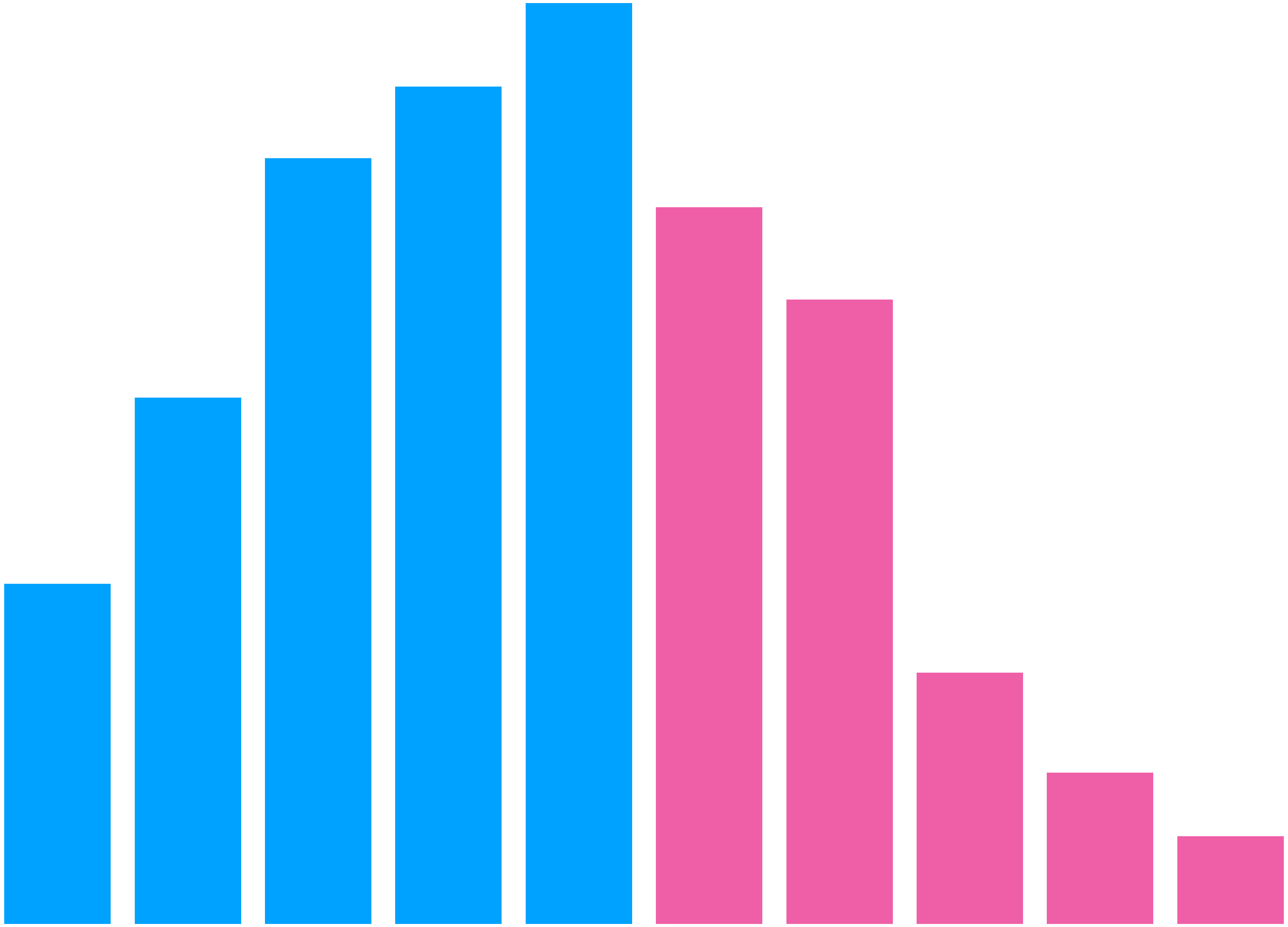
John helped his professor, Professor Reese, to conduct a topographic survey of a piece of land. He walked in a straight line, noting down the elevation of the land at every centimeter. After he is done, he passed the data to Professor Reese. The professor then asked him, "what is the peak elevation of the land?" John did not know the answer! He could write a program to scan through the millions of data points he collected, but he knew that, since you have taken CS1010, you can do a better job. So, John asked for your help.

You first clarify the problem with John: "What is a peak?" To which John explained that a peak is a location that is strictly higher than the surrounding locations. You then asked: "Is it guaranteed that there is exactly one peak?" John then explained the pattern in the data: the elevation always either remains the same or increases as he walks. After he passed the peak, the elevation always either remains the same, or decreases. But he cannot remember if he ever encountered a peak -- it might be possible that the elevations data is always non-decreasing, or non-increasing, or there is a flat plateau where there are multiple highest locations with the same elevation. So, a peak might not exist. But if there is a peak, it is guaranteed that there is exactly one peak.



Question 2: Scripts (10 marks)

Professor Reese is teaching a huge class at the university. He finished grading a test and he asked John to help him enter the grades into IVLE grade book, in increasing order of the student id. John asked the professor, "Are the scripts sorted?", to which the professor answered, "Almost! The top portion of the pile is sorted in increasing order. The rest, in decreasing order." The professor then left after saying "Hasta la vista, baby," leaving John to wonder how to deal with the test scripts. John needed to sort the scripts in increasing order of the student id. So he messaged you to help him figure out an efficient algorithm to do this. "No problemo!", you said, "Can be done in $O(n)$!" You said. So you went ahead and wrote out the following program to show John how he can solve his problem in $O(n)$ time.



Question 2: Fill (15 marks)

Scully is attending a drawing class today. She has already completed her work, but the teacher is not satisfied with her choice of colors and wants her to re-color some parts. She asks you, a programming genius, to help.

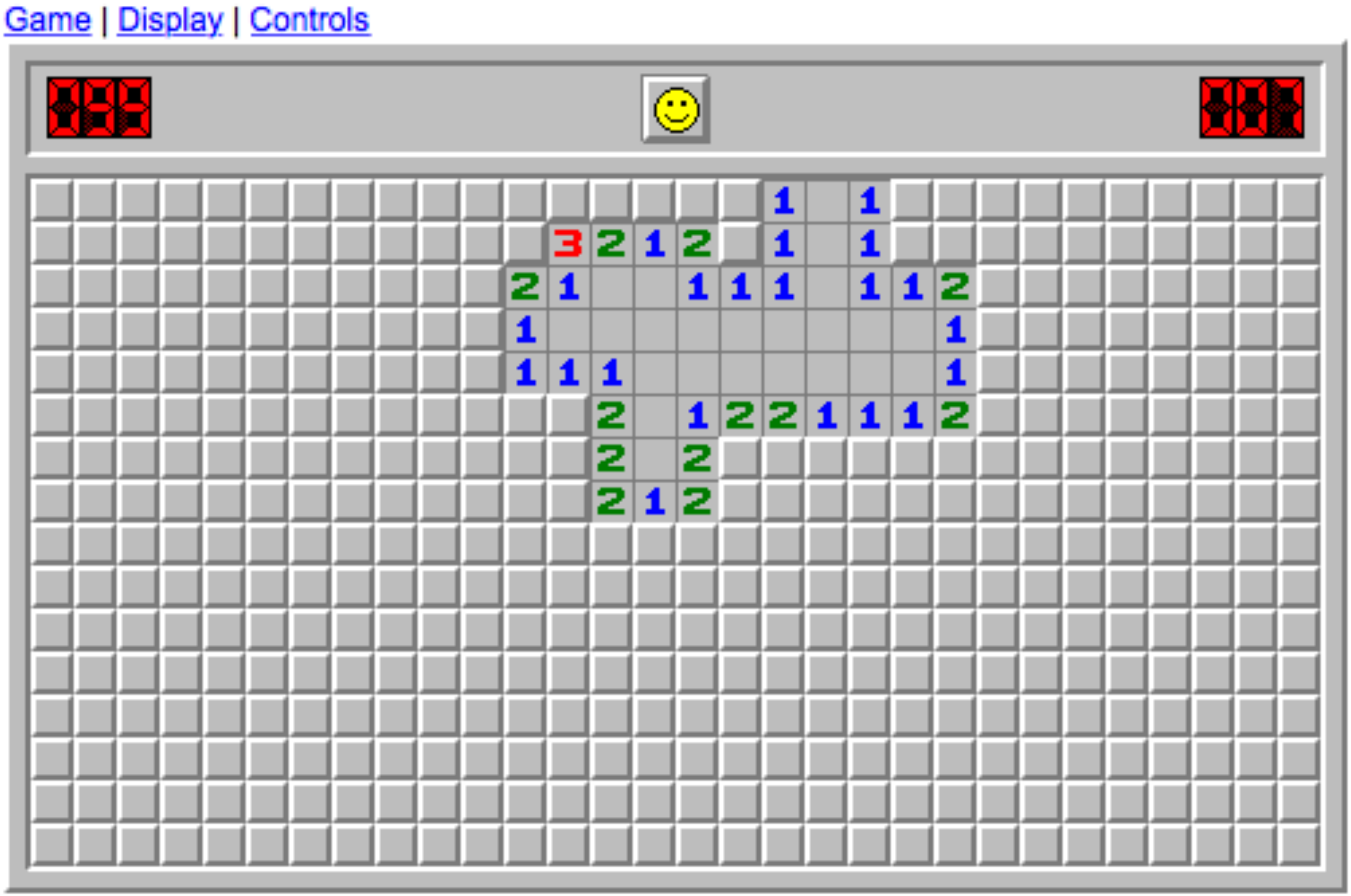
Scully's drawing can be simplified as a 2D array of size $m \times n$, with colors '0' to '9'. The drawing consists of several objects. Each object can be viewed as connected areas of cells of the same color. Two cells are connected if they share a common edge, i.e. a cell will have at most 4 connected cells.

Write a program, `fill.c`, to help Scully re-color her drawing according to her teacher's requirement. It reads from standard input two positive integers m and n in the first line, followed by m lines of strings. Each string is of length n, consisting of only characters '0' to '9'. The next line is a positive integer q, which is the number of color changes the teacher requires. Following this, there are q lines with three integers on each line: x_i, y_i, and c_i. It means to color the object containing pixel (x_i, y_i) to the color c_i. We denote the top left pixel to be $(0, 0)$ and the indices increases towards the right and down.

```
5 5
00110
00110
00000
11040
01100
```

Who draw pictures like this??

abstract
generalize
solve


Game | Display | Controls

Abstraction

identifying and abstracting relevant information

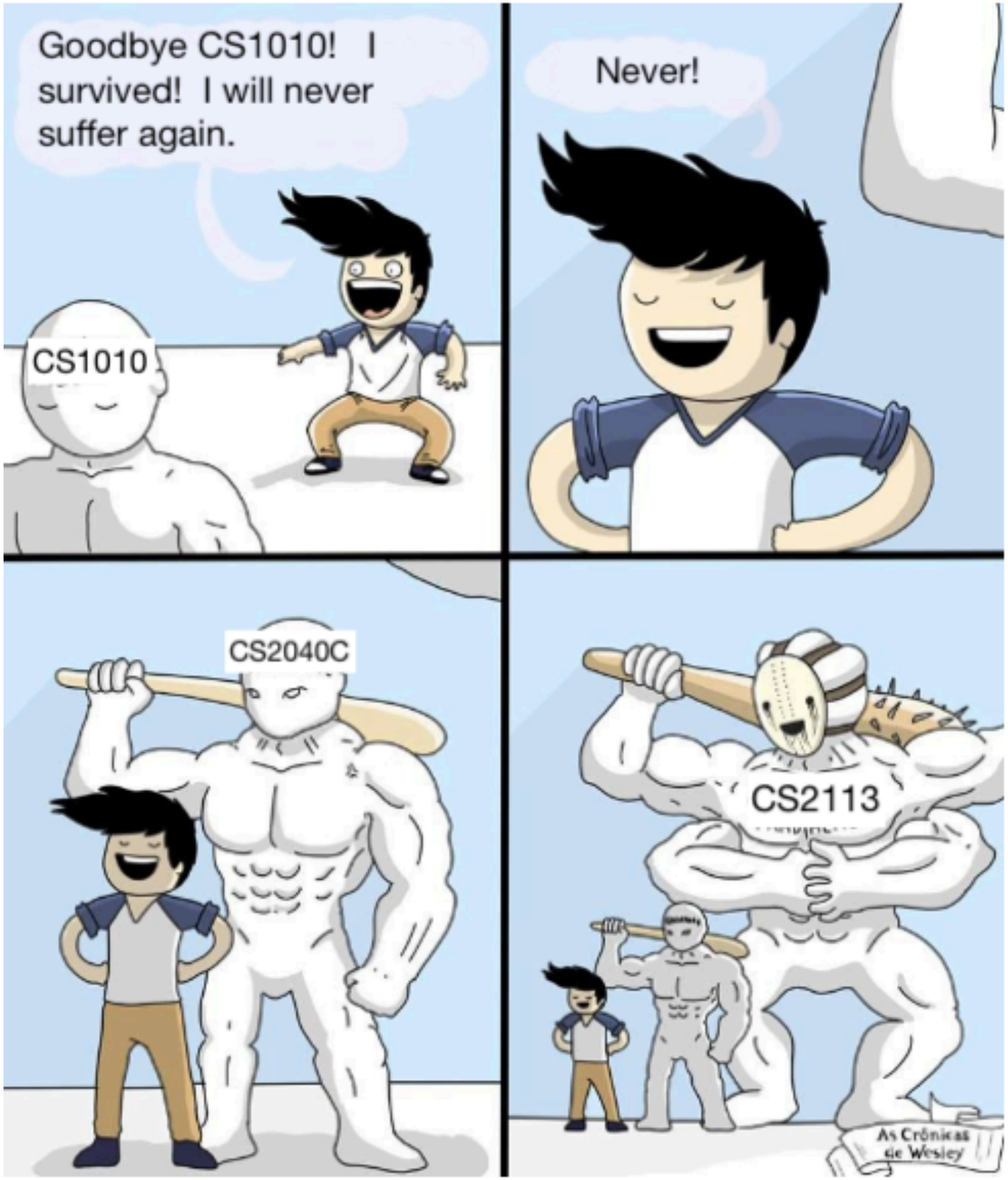
generalize to other domains

1. Decomposition
2. Pattern Recognition
3. Abstraction
4. Algorithms

Some words of advice

Work hard.
Very very hard.

what doesn’t kill you
only makes you
stronger


Goodbye CS1010! I survived! I will never suffer again.
CS1010
Never!
CS2040C
CS2113
As Crônicas de Wesley