Lecture 2A

COMP 1100

Acknowledgement of Country

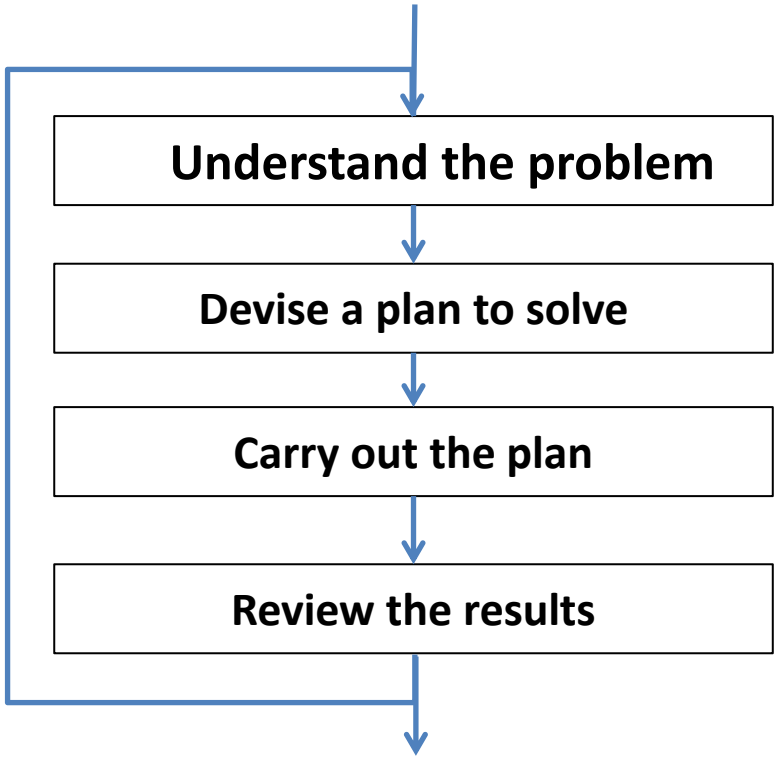
- ✓ *I wish to acknowledge the traditional custodians of the land we are meeting on, the Ngunnawal people. I wish to acknowledge and respect their continuing culture and the contribution they make to the life of this city and this region. I would also like to acknowledge and welcome any other Aboriginal and Torres Strait Islander people who are enrolled in our courses.*

Content

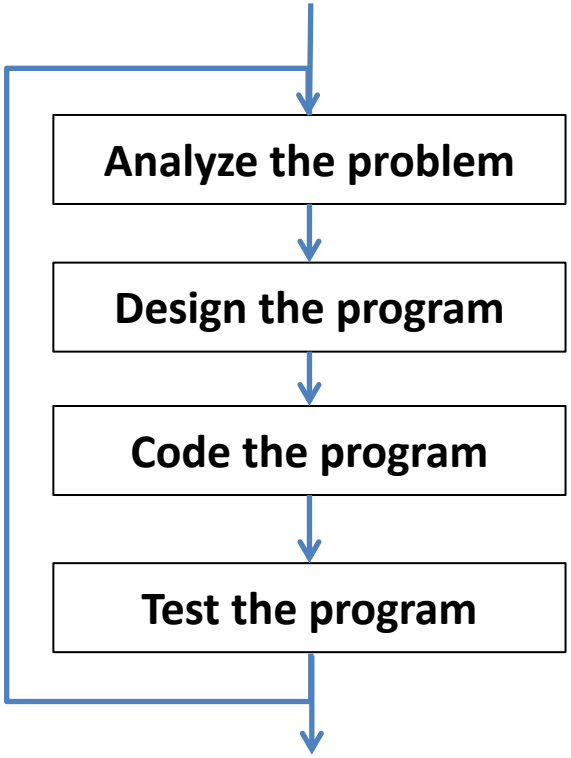
- ✓ Programming as problem solving
- ✓ Sets and functions overview

Programming as Problem Solving

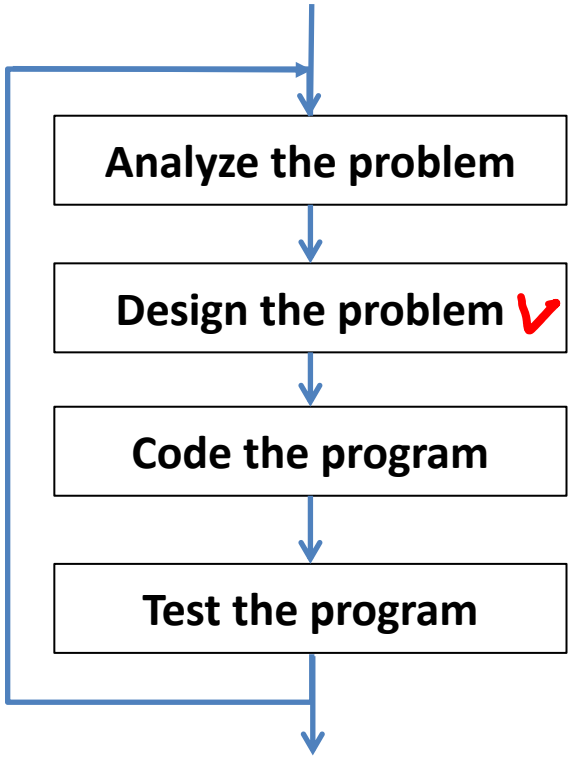
Problem solving principles



Developing a Program



Programming as Problem Solving



1. Specify the problem requirements

Important questions

- ✓ What are the inputs? (given data)
- ✓ What are the outputs? (required data)
- ✓ How will we calculate the required outputs from the given inputs?

2. Create an outline of the program that solves the problem
   - ✓ An **algorithm –** a step by step procedure that will provide the required results from the given inputs.
   - ✓ *Algorithm Examples:*
     - ✓ Instructions on how to make a cake,
     - ✓ How to use the bank's ATM, etc.

3. Once the design is completed, implement the program **code**.

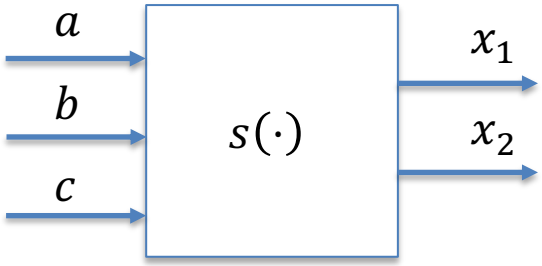
4. Testing is done throughout the development cycle
   - ✓ Mental walkthrough
   - ✓ Ultimate test is to run the program to see if the outputs are correct for the given inputs.

Problem classification

- ✓ A programming language is a notational system for describing computation in a **machine-readable** and **human-readable** form.
- ✓ **Types of problems**
  - ✓ **Functional problems**
  - ✓ **Decision problems**
  - ✓ **Search problems**
  - ✓ **Optimization problems**

Problem examples

- **Polynomial root finding**
  - **Category:** Functional problem
  - **Input:** A polynomial with real coefficients
  - **Output:** One (or all) real roots of the input polynomial
- **Example**: $ax^2 + bx + c = 0$ solver



$s :: (a, b, c) \rightarrow (x_1, x_2)$

where

$$x_1 = \frac{-b - \sqrt{d}}{2a},$$

$$x_2 = \frac{-b + \sqrt{d}}{2a}$$

$$d = b^2 - 4ac$$

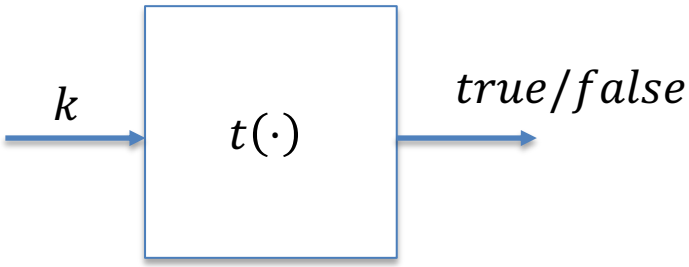
```
-- Quadratic equation solver
-- ax^2 + bx + c = 0
-- INPUT: a, b, c
-- OUTPUT: x1 x2

s :: (Float, Float, Float) -> (Float, Float)
s (a, b, c) = (x1, x2)
```

Is this implementation general enough ?

```
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( Main.hs, interpreted )
Ok, one module loaded.
*Main> s (1,-3,2)
(2.0,1.0)
```

Problem examples

- **Primality testing**
  - **Category:** Decision problem
  - **Input:** A positive integer
  - **Output:** The decision whether the input integer is prime or not
- **Example**: Is k prime?




```
k = 20
[ x | x <- [2..k - 1], mod k x == 0]
[2, 4, 5, 10]
k = 13
[ x | x <- [2..k - 1], mod k x == 0]
[]
```

$$t :: k \rightarrow \{true, false\}$$

$$t = \begin{cases} true, & k \in \mathbb{P} \\ false, \text{otherwise} \end{cases}$$

$$\mathbb{P} = \{k \in \mathbb{N}_2 | \ (\forall m \in \mathbb{N}_2) \ (\text{mo}$$

```
-- Primality testing
t :: Integer -> Bool
t k =
    if k > 1 then
        null [ x | x <- [2..k - 1], mod k x == 0]
    else
        False
```

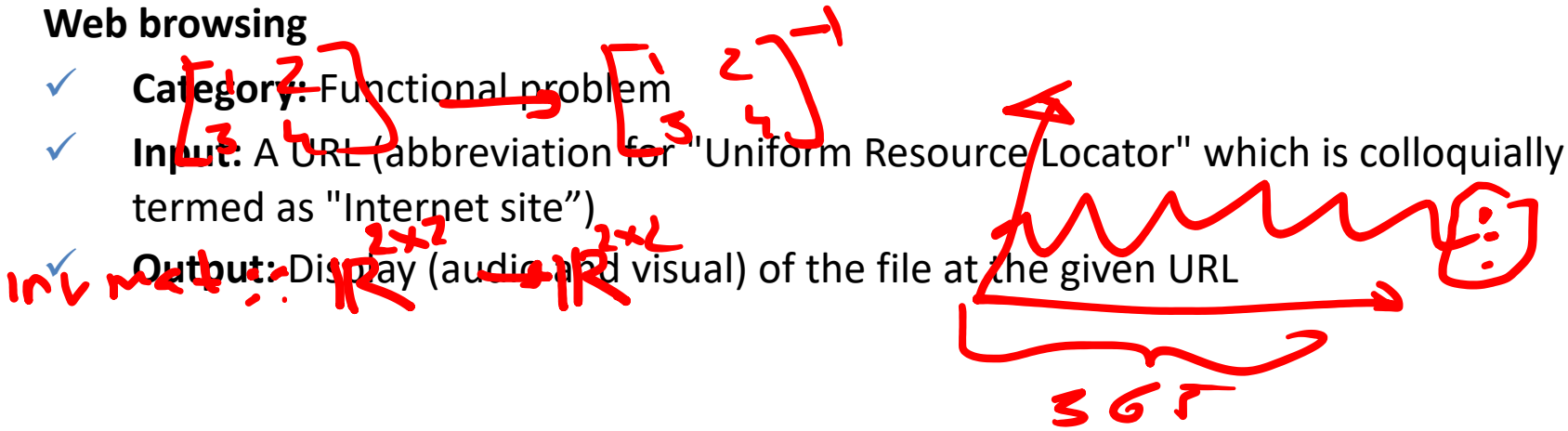
```
*Main> t 10
False
*Main> t 13
True
```

Problem examples

- **Matrix inversion**
  - **Category:** Functional problem
  - **Input:** A square matrix with rational entries
  - **Output:** The inverse of the input matrix if it is invertible, or "failure”

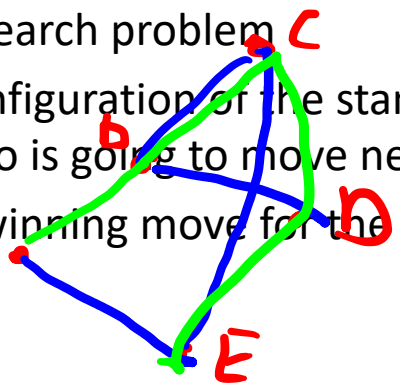
Problem examples

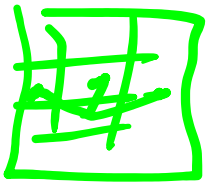
- **Weather prediction**
  - **Category:** Functional problem
  - **Input:** Records of weather for previous days and years. Possibly also data from satellites.
  - **Output:** Expected weather of Canberra for tomorrow
- **Web browsing**
  - **Category:** Functional problem
  - **Input:** A URL (abbreviation for "Uniform Resource Locator" which is colloquially termed as "Internet site")
  - **Output:** Display (audio and visual) of the file at the given URL



Problem examples

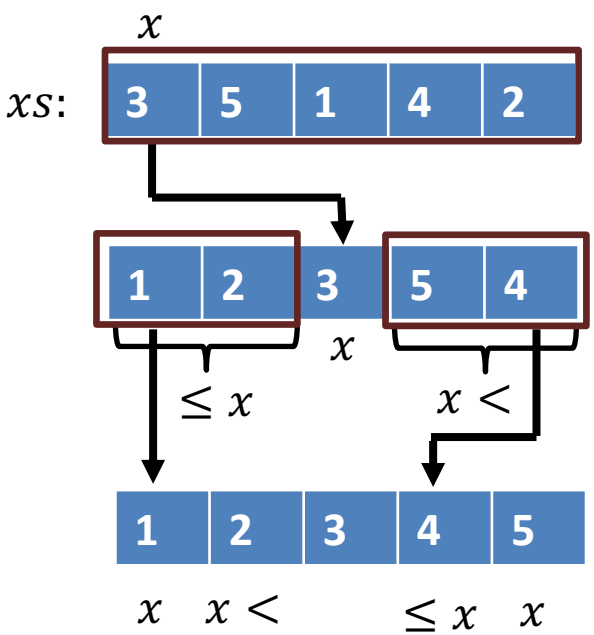
- **Traveling salesman problem (TSP)**
  - **Category:** Optimization problem
  - **Input:** A set of cities, the cost of traveling between each pair of cities, and the criterion of cost minimization
  - **Output:** A route through all the cities with each city visited only once and with the total cost of travel as small as possible
- **Chess : Can I win?**
  - **Category:** Search problem
  - **Input:** A configuration of the standard 8x8 chess board and the player ("white" or "black") who is going to move next
  - **Output:** A winning move for the next player, if existent, or "failure"

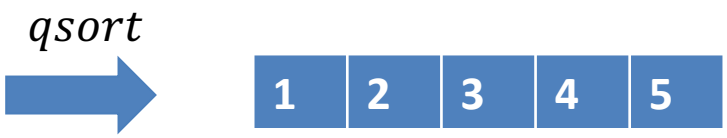




Problem examples

- **Sorting**
  - **Category:** Functional problem
  - **Input:** A list of elements
  - **Output:** An ordered list of elements
- *Example*:






```
qsort :: [Integer] -> [Integer]
qsort [] = []
qsort (x:xs) = qsort smaller ++ [x] ++ qsort larger
    where
        smaller = [a | a <- xs, a <= x]
        larger  = [b | b <- xs, b > x]
```

```
*Main> qsort [3, 5, 1, 4, 2]
[1,2,3,4,5]
```

QUIZ

- Come up with
  1. a search problem,
  2. a functional problem,
  3. an optimization problem,
  4. a decision problem.

  So overall present 4 different problems.

  For each problem specify

  a. Inputs
  b. Outputs
  c. If possible briefly discuss how inputs are converted to outputs (i.e. give a brief description of the algorithm)