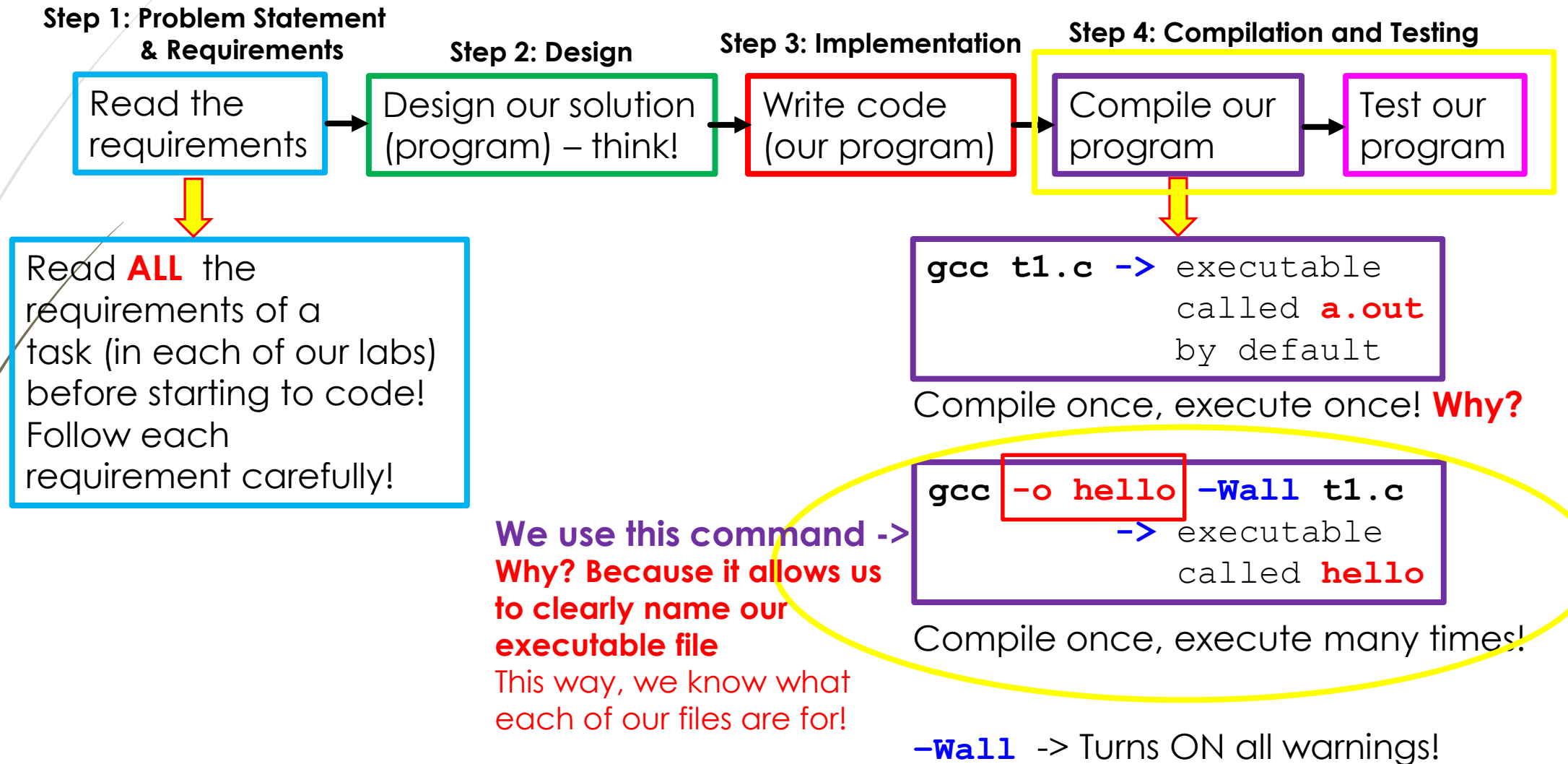# Lab 1

Helpful Tips

Welcome to Lab 1!

- Lab 1 is long (> 3 hours)
  - You will need to finish it outside your lab session
- Do not skip ahead to the tasks!

  - **Read everything!**

For all labs – all tasks:

- Always start your lab
  - **by remotely logging into CSIL**
  - by `cd` to from **your local repo directory**:

    `sfuhome/cmpt-127/cmpt127-1211-your_username`

In CMPT 127 we shall learn: Software development process

Step 1: Problem Statement & Requirements

Step 2: Design

Step 3: Implementation

Step 4: Compilation and Testing

Read the requirements	Design our solution (program) – think!	Write code (our program)	Compile our program	Test our program

Read **ALL** the requirements of a task (in each of our labs) before starting to code! Follow each requirement carefully!

```
gcc t1.c -> executable
            called a.out
            by default
```

Compile once, execute once! **Why?**

We use this command ->
Why? Because it allows us to clearly name our executable file
This way, we know what each of our files are for!

```
gcc -o hello -Wall t1.c
        -> executable
           called hello
```

Compile once, execute many times!

-Wall -> Turns ON all warnings!

A Few Helpful Tips for some Lab 1 Tasks

- **In general, tasks of Lab 1** build on each other

  - When doing a task, ask yourself: what code construct can I recycle from the previous task?

- **Be aware: Task 5** and **Task 7** - Input and output interleaved on the monitor screen

- **Task 6** – There is a trick that allows us to properly initialize variables `max` and `min`

  - Trick: Initialize `max` to the smallest possible value the input can have and `min` to the largest possible value the input can have

- Read **Task 9**'s Guide carefully - Also see Task 9 Clarification

C Functions

For all labs – all tasks:

- Get in the habit of looking up C functions (introduced in labs) on the Internet
  - Look for:
    - Purpose of the function
    - Parameters (numbers, data types and order) of the function
    - Returned value of the function
      - For example, what does it mean when `scanf(…)` returns a number that is …

```
> 0? ->  ???
  0? ->  ???
< 0? ->  ???
```

for you
to answer