EE538: Computing Principles for Electrical Engineers

University of Southern California

Instructor: Arash Saifhashemi

Who Is This Course For?

Preparing you for **design**, **analysis**, and **implementing** a **complete** software **system**

Two unit courses:

- C++, **Algorithm**, and **Data Structure**

Who Is This Course For?

Do not take this course if you are only looking to take an easy two-unit course!

EE538 involves several **homework assignments**, a **project**, a lot of **coding**, and studying **various algorithms**.

I really encourage you to **take the course seriously** and plan to dedicate enough time to complete the homework assignments and the project.

About me:

- Very Trojan!
  - Student: USC EE: 2004 to 2012
  - Lecturer since: Jan 2020
- Worked at both big and small corporations
  - Hardware and software
- Startups
  - Mobile apps, full stack
- Guess what language I used in all of them?

Introduction

- Coding
  - Basics of C++
- Foundations of Software Engineering
  - Testing, Source Control, Shell scripts
  - Modular Programming
  - Object Oriented Programming
- Basics of Algorithms and Data Structure
  - Trees, linked-lists, hash tables, heaps, ...
  - Runtime analysis
  - Algorithm design and analysis
    - Greedy, recursive, dynamic programming, …

Books

- Algorithm, 4th Edition (required) Robert Sedgewick, Kevin Wayne (available at the campus store)
- Algorithms in C++, 3rd Edition (optional, C++ supplement to 1), Robert Sedgewick, Kevin Wayne (available at the campus store)
- The C++ Programming Language, 4th Edition (recommended) Bjarne Stroustrup (available at the campus store)
- Code Complete: A Practical Handbook of Software Const

Videos and Optional Material

- All lecture recordings will be provided on Blackboard’s Zoom section.
- Occasionally, I will send you **optional videos** to watch.
  - Watching these videos are completely optional, and will not affect your grades.
  - There will be no questions from the extra material in these videos.
- Optional questions in homework assignments:
  - They will not be graded.
  - We may not provide solutions.

Final Grades

Assessment Tool (assignments)	**% of Grade**
Homework	35%
Project	25%
Exam #1	20%
Exam #2	20%
TOTAL	100

Course Platform

- Repos we use:
  - https://github.com/ourarash/cpp-template
  - https://github.com/ourarash/cpp_tour
- Piazza
  - All the lecture material will be posted here!
  - **Homework assignment will be released here!**
- GitHub classroom
  - **Submit your homework through GitHub before the deadline!**
- Blackboard/DEN
  - **Your scores will be posted here!**

Piazza

- All sections will share the same **Piazza** link.
- Everyone should've received an email invite to the Piazza for this course
  - **All** questions about assignments, course material, exams, etc. should go there
  - Please read this before asking questions.
- Send an email if:
  - **Regrade question** on Programming Assignment (email the TA who graded you)
  - **Personal question** (such as DSP, etc.)
- **Try** to generalize questions so that they can be public (visible to everyone)
  - ..**BUT** don't put more than 5 lines of code in a public post
- If you need more code than that for context, use a private post
- Bad questions:
  - "here's lots of code, fix it!" cries for help will be ignored – you need to demonstrate that you've actually tried to debug it yourself!

Other Places to Look

Have you heard about this thing called:

- Google Search?
- Stackoverflow?

HW Late Policy

- 1 day late is -15 points
- 2 days late is -30 points
- 3 days late is -45 points

More than 3 days late is a 0

In case of emergency, we will decide on a case by case basis if an extension is warranted. You may need to show documentation depending on the extension request.

Usually assignments have 120 - 130 points, 100 points is considered full credit.

Academic Integrity

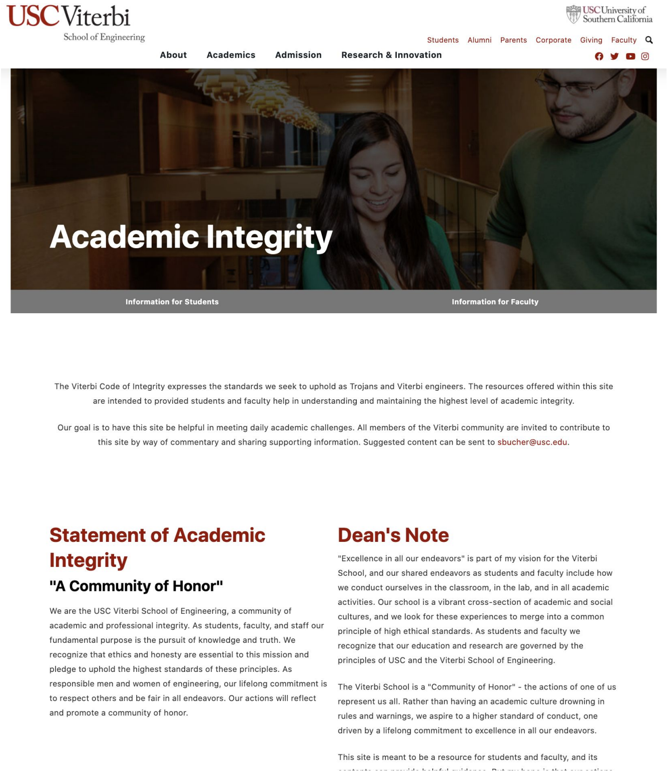
- All assignments should be your **individual work**
- If you are asking classmates questions beyond the scope of what you could reasonably ask on Piazza in a “public” post, then you are not doing individual work
- **Do not** share your code files or part of your code files with your classmates (current or future students)
- **Do not** post your code on a publicly-accessible website (GitHub, course hero, etc).
- **Do not** step through anyone else’s code, if you need help debugging ask an instructor or TA or Piazza
- If you’re not sure if something is allowed: ask an instructor. Instructors/TAs are always happy to help!

The most important thing - Academic Integrity

- University Policy
  - https://sjacs.usc.edu/students/academic-integrity/
  - https://viterbischool.usc.edu/academic-integrity/

We hope you could...

- Be responsible for yourself!
- Learn some useful knowledge!



The Viterbi Code of Integrity expresses the standards we seek to uphold as Trojans and Viterbi engineers. The resources offered within this site are intended to provided students and faculty help in understanding and maintaining the highest level of academic integrity.

Our goal is to have this site be helpful in meeting daily academic challenges. All members of the Viterbi community are invited to contribute to this site by way of commentary and sharing supporting information. Suggested content can be sent to sbucher@usc.edu.

Statement of Academic Integrity

"A Community of Honor"

We are the USC Viterbi School of Engineering, a community of academic and professional integrity. As students, faculty, and staff our fundamental purpose is the pursuit of knowledge and truth. We recognize that ethics and honesty are essential to this mission and pledge to uphold the highest standards of these principles. As responsible men and women of engineering, our lifelong commitment is to respect others and be fair in all endeavors. Our actions will reflect and promote a community of honor.

Dean's Note

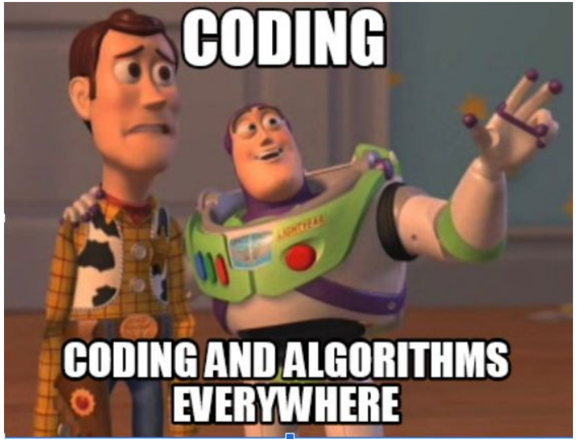
"Excellence in all our endeavors" is part of my vision for the Viterbi School, and our shared endeavors as students and faculty include how we conduct ourselves in the classroom, in the lab, and in all academic activities. Our school is a vibrant cross-section of academic and social cultures, and we look for these experiences to merge into a common principle of high ethical standards. As students and faculty we recognize that our education and research are governed by the principles of USC and the Viterbi School of Engineering.

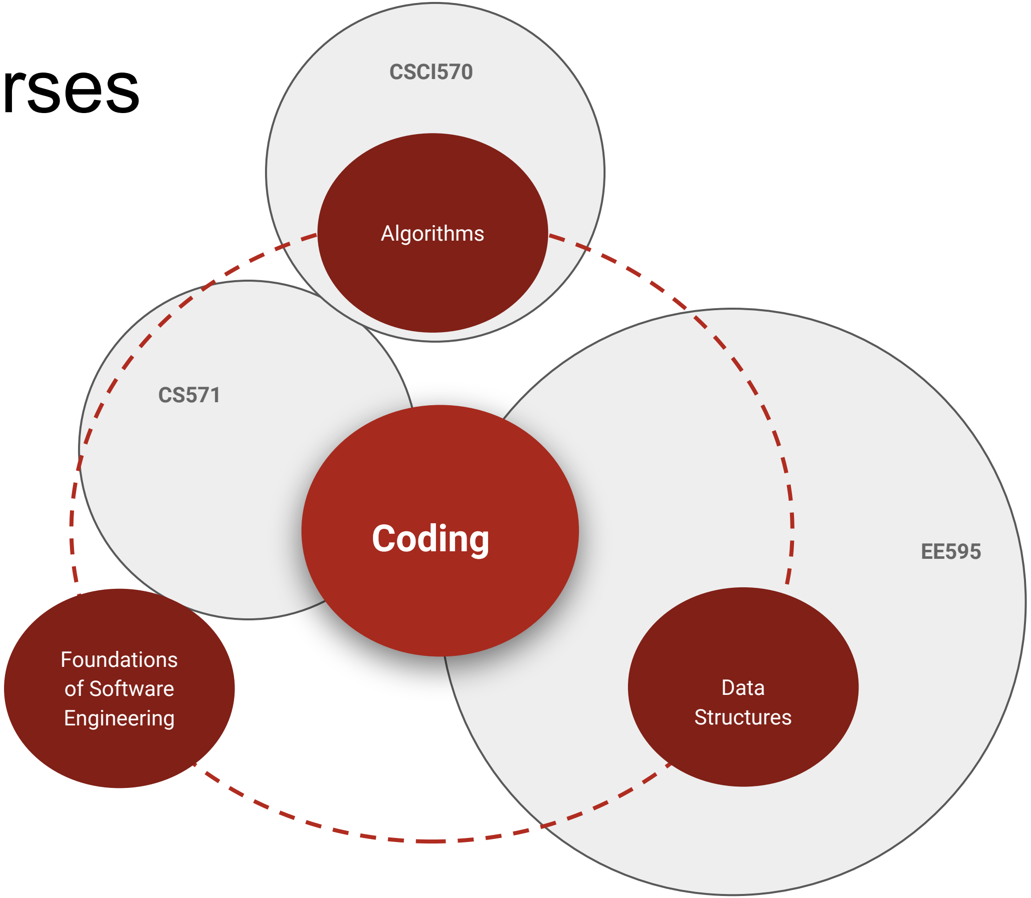
The Viterbi School is a "Community of Honor" - the actions of one of us represent us all. Rather than having an academic culture drowning in rules and warnings, we aspire to a higher standard of conduct, one driven by a lifelong commitment to excellence in all our endeavors.

This site is meant to be a resource for students and faculty, and its

Overlap with Other Courses

- We are focused on coding:
    - Almost all algorithms discussed should be implemented and tested.





This is a coding-oriented course!

Programmer Basic Tools

- We should all have:
  - **Stackoverflow** account
  - **Github** account
- Please make sure you install:
  - Visual Studio Code
  - Git
  - Bazel
  - Linux-compatible terminal
  - C++ Toolchain
    - Mac: Xcode
    - Windows:
      - Preferred: Use VirtualBox
      - Cywin or MinGW
    - Linux: g++

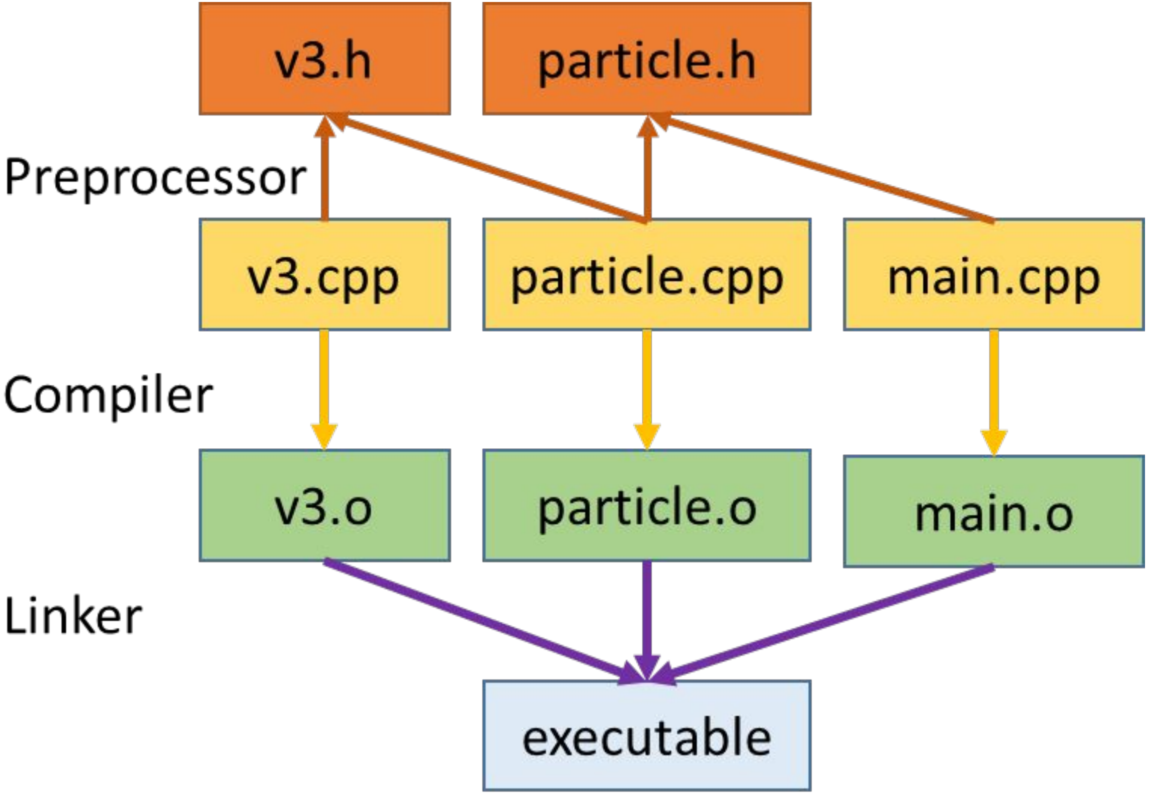
We will use C++

Python	C++
Easier to learn	**Harder** for beginners
Interpreted	Compiled
Slow	Very **fast**
Lots of libraries	Libraries are **harder** to access
Machine learning and data science	**Speed**: Gaming, Device Drivers, Servers, Stock Market

Where is C++ Used?

Ai
Adobe Illustrator
Google
chrome
Windows
Mac OS
mozilla Firefox
MySQL
Ps PHOTOSHOP
Office

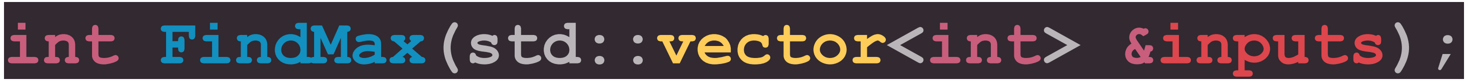
C++ Needs a Compiler



Example

Find the maximum value in an array of integers

- **Step 1:** Clearly define the input and output of the problem



```
int FindMax(std::vector<int> &inputs);
```

Example

Find the maximum value in an array of integers

Let’s assume all numbers are higher than -1.

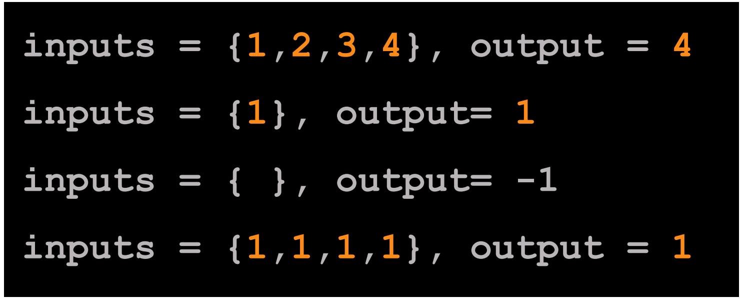
- **Step 2:** What are some example input/outputs?
    - Find corner cases

```
inputs = {1,2,3,4}, output = 4

inputs = {1}, output= 1

inputs = { }, output= -1

inputs = {1,1,1,1}, output = 1
```

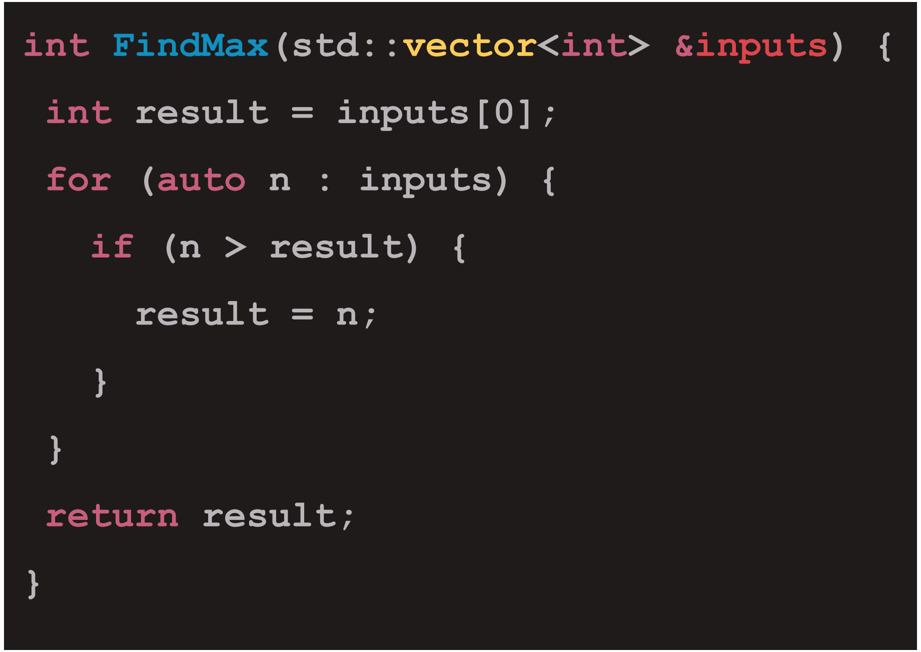
Example

Find the maximum value in an array of integers

- **Step 3:** Propose an algorithm
  - May not be perfect.
  - We will learn various techniques in this course



```
inputs = {1,2,3,4}, output = 4
inputs = {1}, output= 1
inputs = { }, output= -1
inputs = {1,1,1,1}, output = 1
```



```cpp
int FindMax(std::vector<int> &inputs) {
 int result = inputs[0];
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

Example

Find the maximum value in an array of integers

- **Step 4**: Test/Debug your algorithm

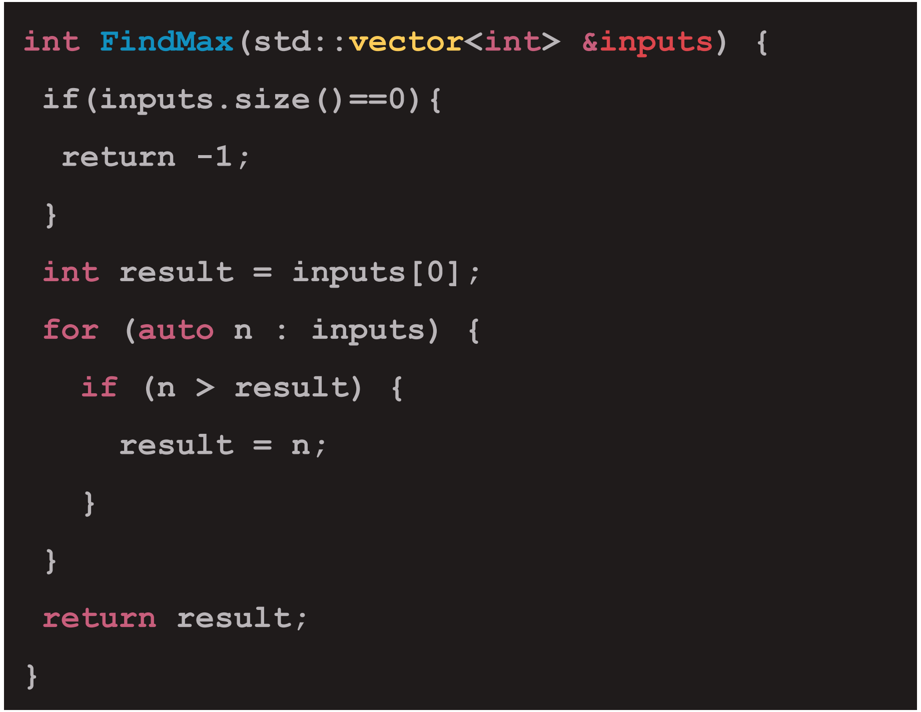
```
int FindMax(std::vector<int> &inputs) {
 int result = inputs[0];
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

```
inputs = {1,2,3,4}, output = 4
inputs = {1}, output= 1
inputs = { }, output= -1
inputs = {1,1,1,1}, output = 1
```

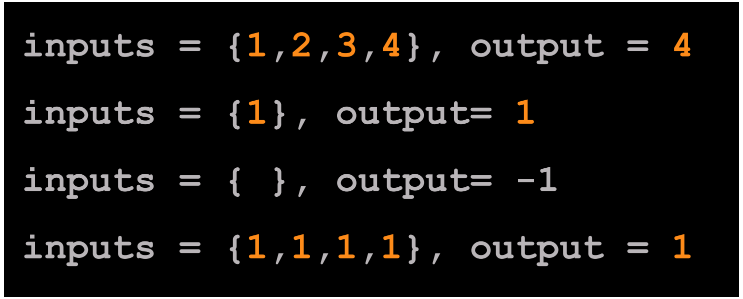
Example

Find the maximum value in an array of integers

- **Step 4**: Test/Debug your algorithm



```
int FindMax(std::vector<int> &inputs) {
 if(inputs.size()==0){
  return -1;
 }
 int result = inputs[0];
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

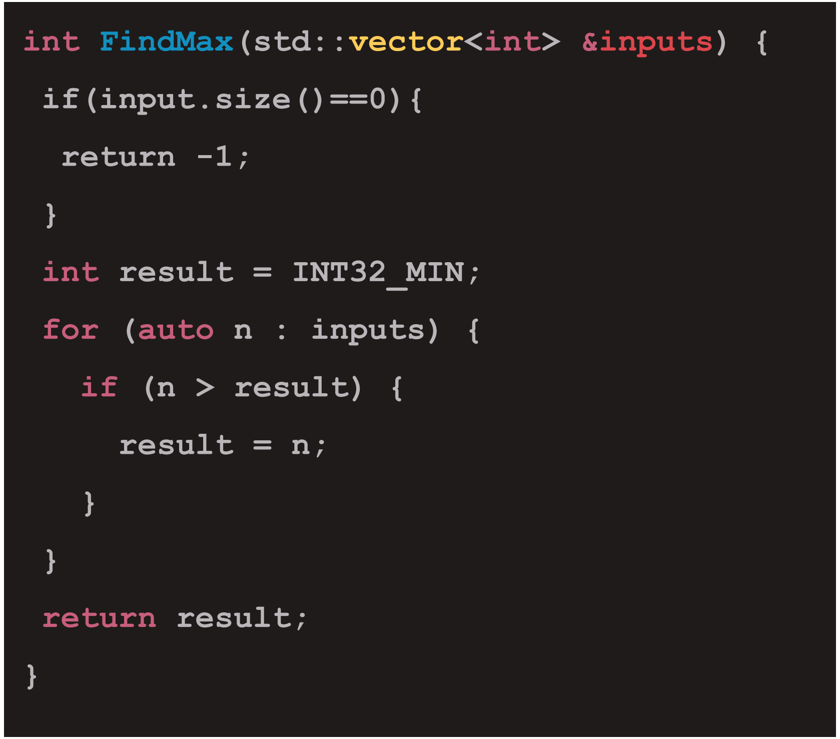


```
inputs = {1,2,3,4}, output = 4
inputs = {1}, output= 1
inputs = { }, output= -1
inputs = {1,1,1,1}, output = 1
```

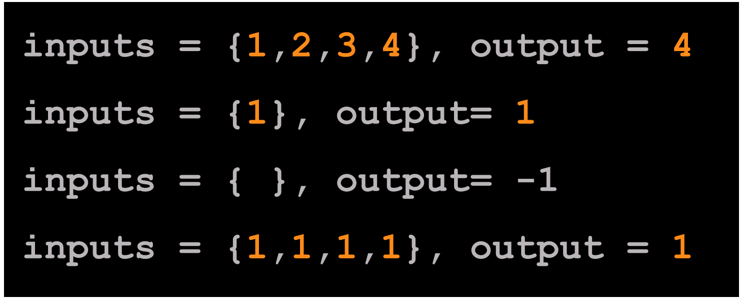
Example

Find the maximum value in an array of integers

- **Step 4**: Test/Debug your algorithm



```
int FindMax(std::vector<int> &inputs) {
 if(input.size()==0){
  return -1;
 }
 int result = INT32_MIN;
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```



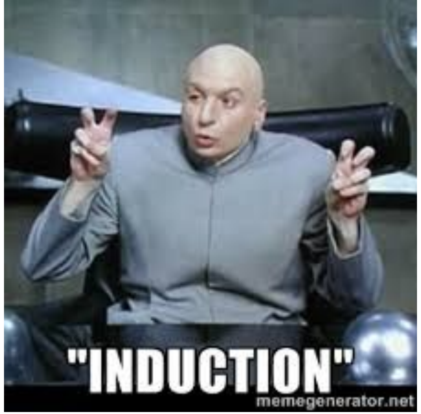
```
inputs = {1,2,3,4}, output = 4
inputs = {1}, output= 1
inputs = { }, output= -1
inputs = {1,1,1,1}, output = 1
```

result: -Inf, 1, 2, .., 4

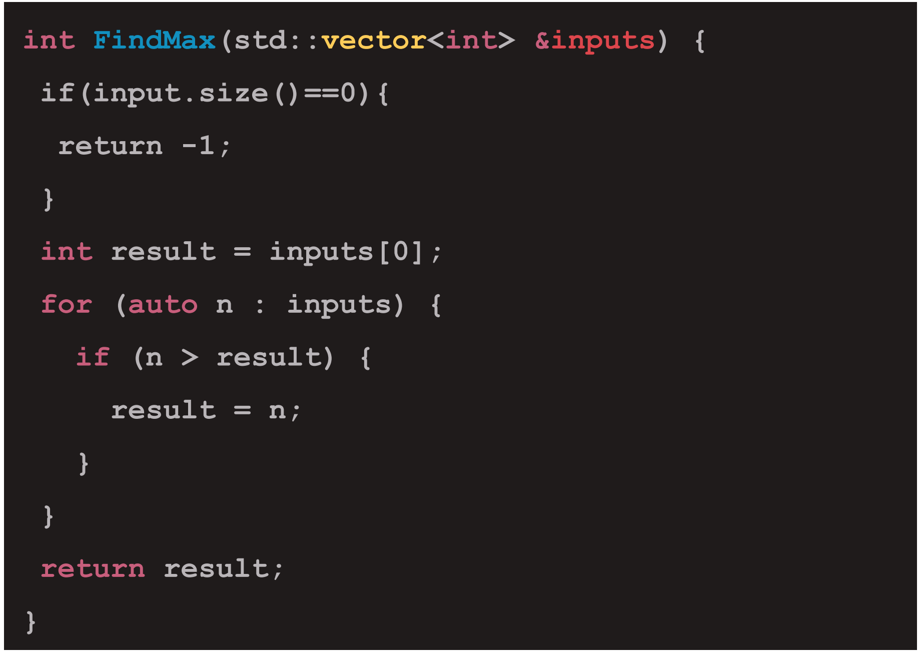
n:1, 2, , ..4

Example

Find the maximum value in an array of integers



- **Step 5**: Prove its correctness
  - Induction
  - Contradiction
  - Case Analysis
  - Other techniques

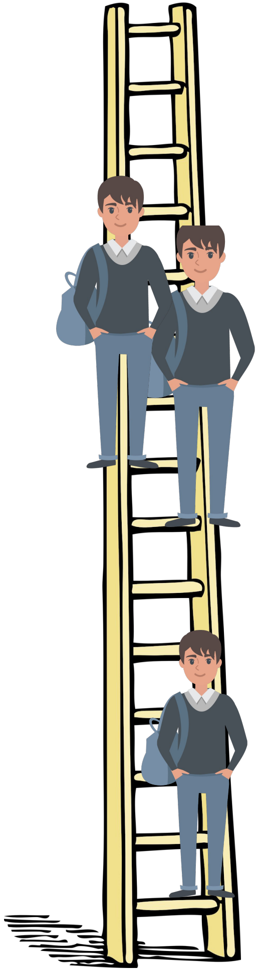


```
int FindMax(std::vector<int> &inputs) {
 if(input.size()==0){
  return -1;
 }
 int result = inputs[0];
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

Proof By Induction

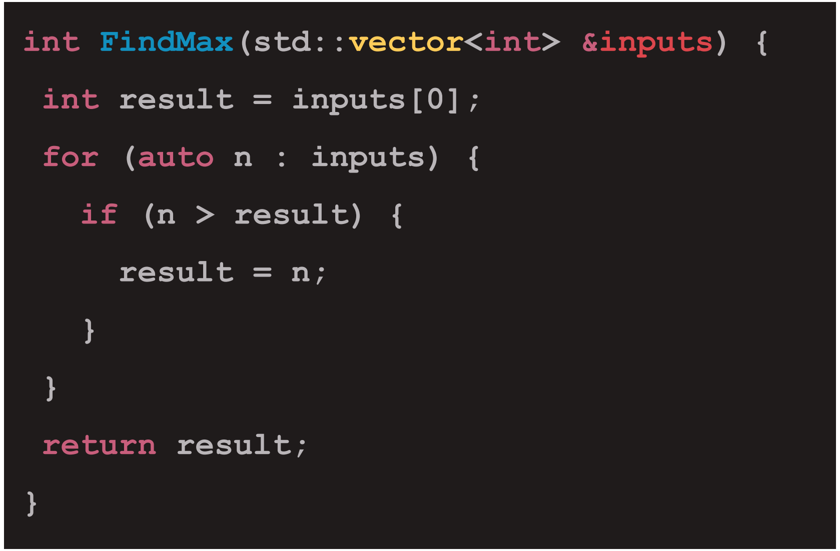
- (Base Case) Show the statement is true for k=1.

- (Inductive Step) Show that if the statement is true for k, this implies the statement is true for k+1.



Example

Find the maximum value in an array of integers



```cpp
int FindMax(std::vector<int> &inputs) {
 int result = inputs[0];
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

- Step 5:
  - Proof by Induction
  - Proof by Contradiction

Proof by Induction: We prove the value of *result* at step i is the max of elements 0 to i.

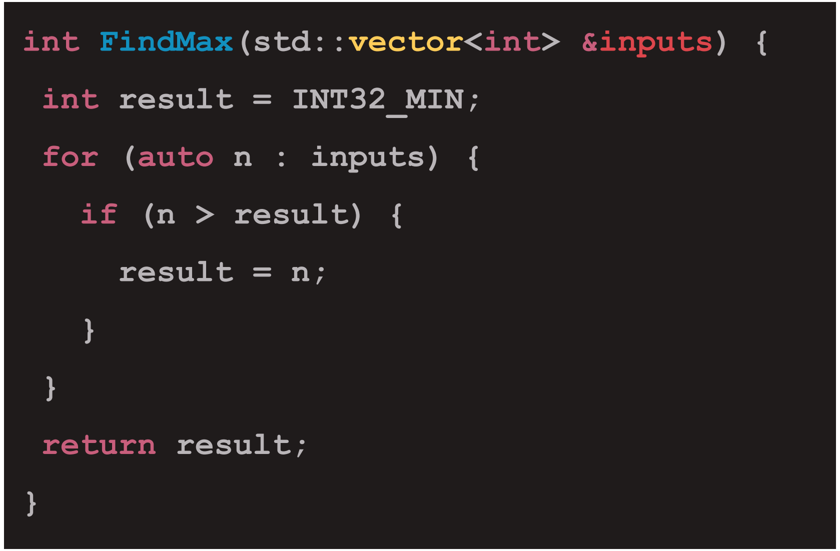
Base case: i = 0: the first time the loop executes.

- Inductive step:
  - *result* is max of [0, …, i], can we say result will be updated to max of [0, …, i+1] ?
  - Example 1: input = [11, 1, 4, 9, 10, 8]
    - 0 to i is: [11, 1, 4, 9] , result = 11, for i+1, the input is [11, 1, 4, 9, 10], n = 10, result stays at 11, and 11 is still maximum!
  - Example 2: input = [11, 1, 4, 9, 12, 8]
    - 0 to i is: [11, 1, 4, 9] , result = 11, for i+1, the input is [11, 1, 4, 9, 12], n = 12, result becomes 12, and 12 is the maximum!

Example

Find the maximum value in an array of integers

- **Step 6:** Systematic Testing

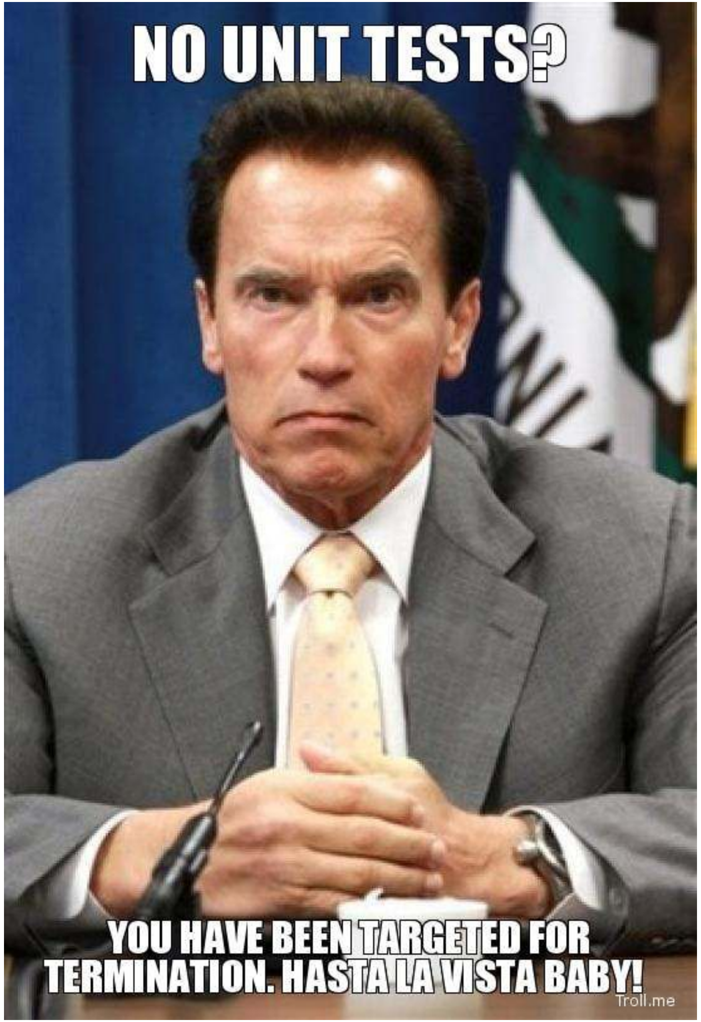


```
int FindMax(std::vector<int> &inputs) {
 int result = INT32_MIN;
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

Why ?

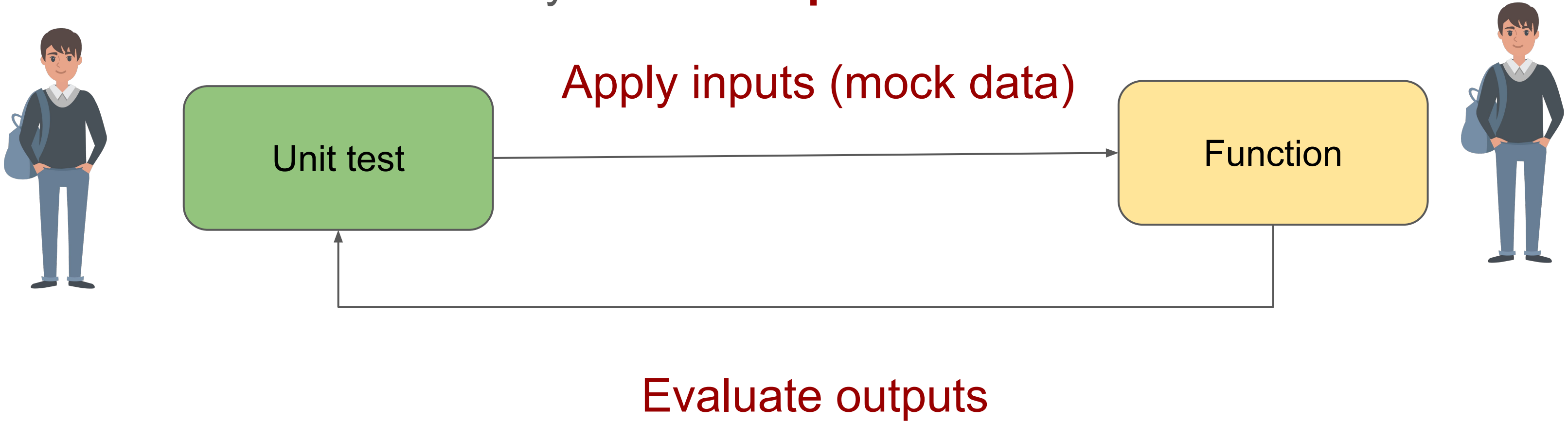
- Even with mathematical proof, there might be **implementation bugs**
- In practice, we may not be able to mathematically prove the correctness

Unit Tests


NO UNIT TESTS?
YOU HAVE BEEN TARGETED FOR TERMINATION. HASTA LA VISTA BABY!
Troll.me

Unit Tests

- *Test your code. Leave nothing to luck!*
- A unit test is a piece of code that tests a function or a class
- Unit tests are written by the **developer!**



Unit test

Apply inputs (mock data)

Function

Evaluate outputs

Google Test Platform

- A testing framework for C++ code
- Automates various tasks:
    - Creates a main function
    - Calls our function under test
    - Applies inputs
    - Provides various functions for testing

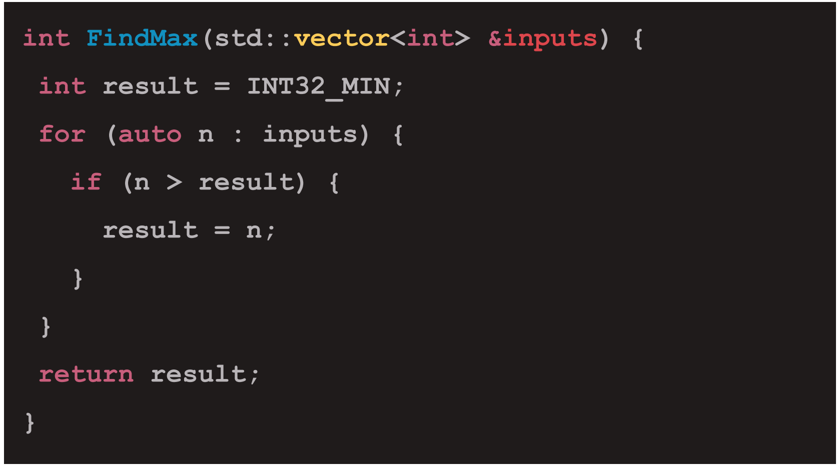
```
// Tests factorial of 0.
TEST(FactorialTest, HandlesZeroInput) {
  EXPECT_EQ(Factorial(0), 1);
}

// Tests factorial of positive numbers.
TEST(FactorialTest, HandlesPositiveInput) {
  EXPECT_EQ(Factorial(1), 1);
  EXPECT_EQ(Factorial(2), 2);
  EXPECT_EQ(Factorial(3), 6);
  EXPECT_EQ(Factorial(8), 40320);
}
```

Fatal assertion	Nonfatal assertion	Verifies
`ASSERT_TRUE(condition);`	`EXPECT_TRUE(condition);`	`condition` is true
`ASSERT_FALSE(condition);`	`EXPECT_FALSE(condition);`	`condition` is false

- `ASSERT_*` yields a fatal failure and returns from the current function.
- `EXPECT_*` yields a nonfatal failure, allowing the function to continue running.

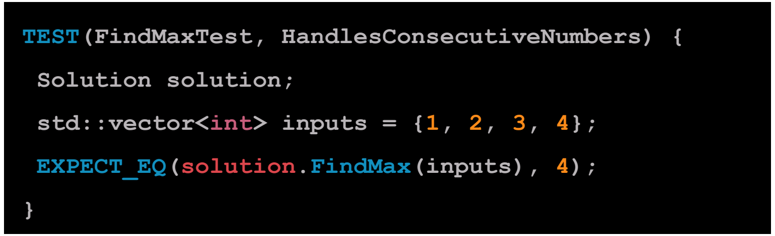
Example

Find the maximum value in an array of integers

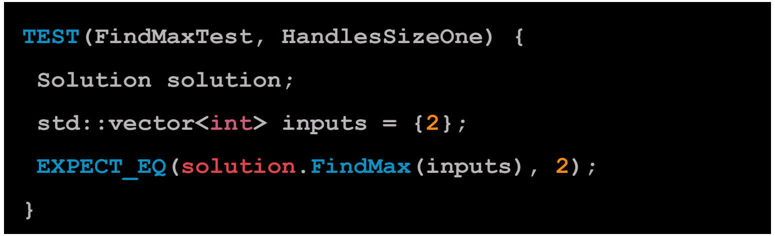


```
int FindMax(std::vector<int> &inputs) {
 int result = INT32_MIN;
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

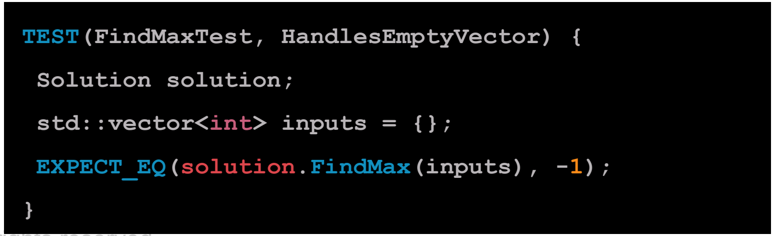
- Step 6: Using Google Test



```
TEST(FindMaxTest, HandlesConsecutiveNumbers) {
 Solution solution;
 std::vector<int> inputs = {1, 2, 3, 4};
 EXPECT_EQ(solution.FindMax(inputs), 4);
}
```

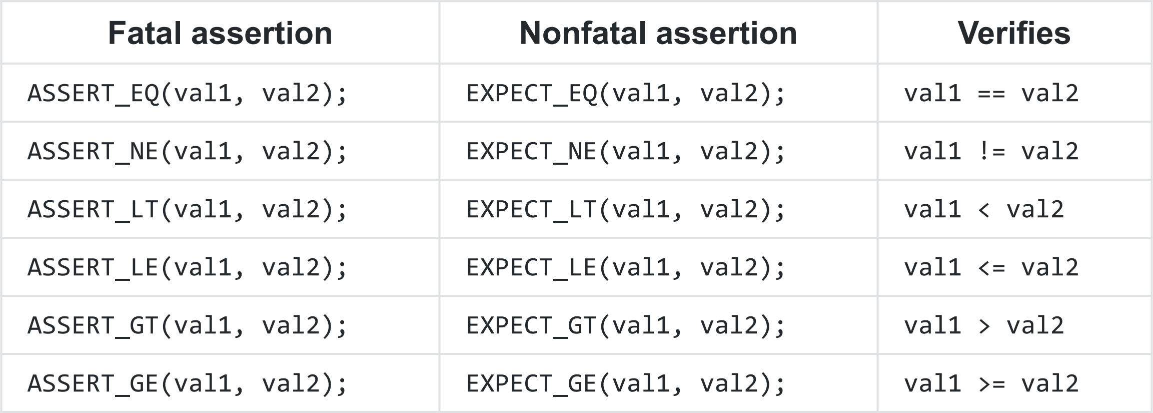


```
TEST(FindMaxTest, HandlesSizeOne) {
 Solution solution;
 std::vector<int> inputs = {2};
 EXPECT_EQ(solution.FindMax(inputs), 2);
}
```



```
TEST(FindMaxTest, HandlesEmptyVector) {
 Solution solution;
 std::vector<int> inputs = {};
 EXPECT_EQ(solution.FindMax(inputs), -1);
}
```

Google Test Platform

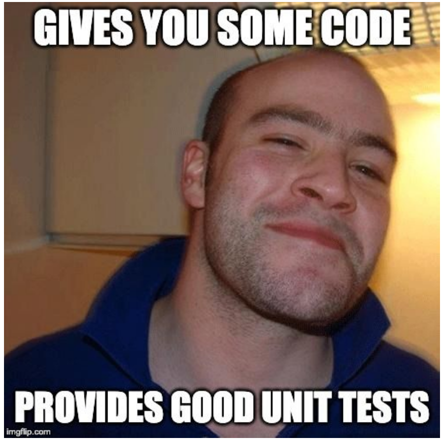


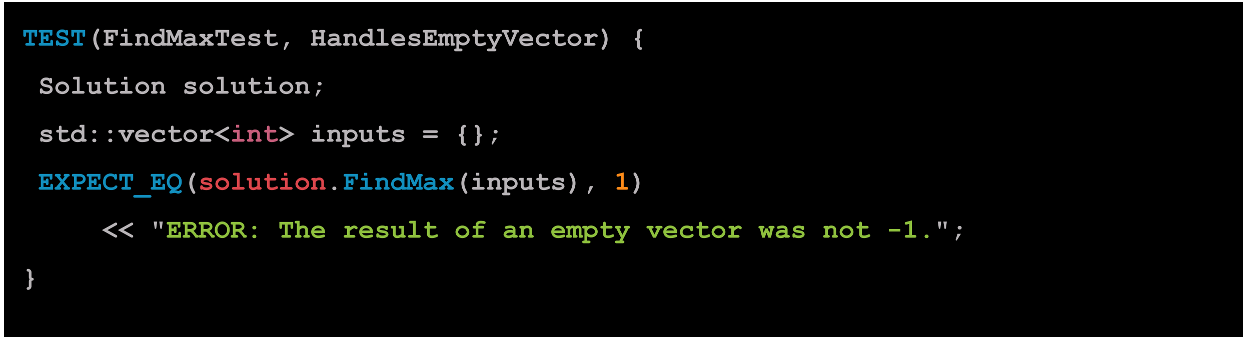
Fatal assertion	Nonfatal assertion	Verifies
`ASSERT_EQ(val1, val2);`	`EXPECT_EQ(val1, val2);`	`val1 == val2`
`ASSERT_NE(val1, val2);`	`EXPECT_NE(val1, val2);`	`val1 != val2`
`ASSERT_LT(val1, val2);`	`EXPECT_LT(val1, val2);`	`val1 < val2`
`ASSERT_LE(val1, val2);`	`EXPECT_LE(val1, val2);`	`val1 <= val2`
`ASSERT_GT(val1, val2);`	`EXPECT_GT(val1, val2);`	`val1 > val2`
`ASSERT_GE(val1, val2);`	`EXPECT_GE(val1, val2);`	`val1 >= val2`

https://github.com/google/googletest

Unit Tests

- Test should be independent and repeatable.
  - A test should not succeed or fail as a result of other tests.
- Tests should be portable and reusable.
  - They should work on different platforms
- Tests should be fast.
- Test should provide as much information about the problem as possible.

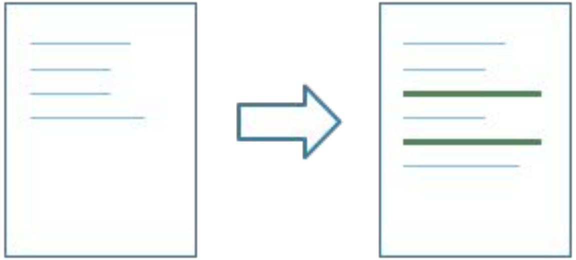




```
TEST(FindMaxTest, HandlesEmptyVector) {
 Solution solution;
 std::vector<int> inputs = {};
 EXPECT_EQ(solution.FindMax(inputs), 1)
     << "ERROR: The result of an empty vector was not -1.";
}
```

Version Control

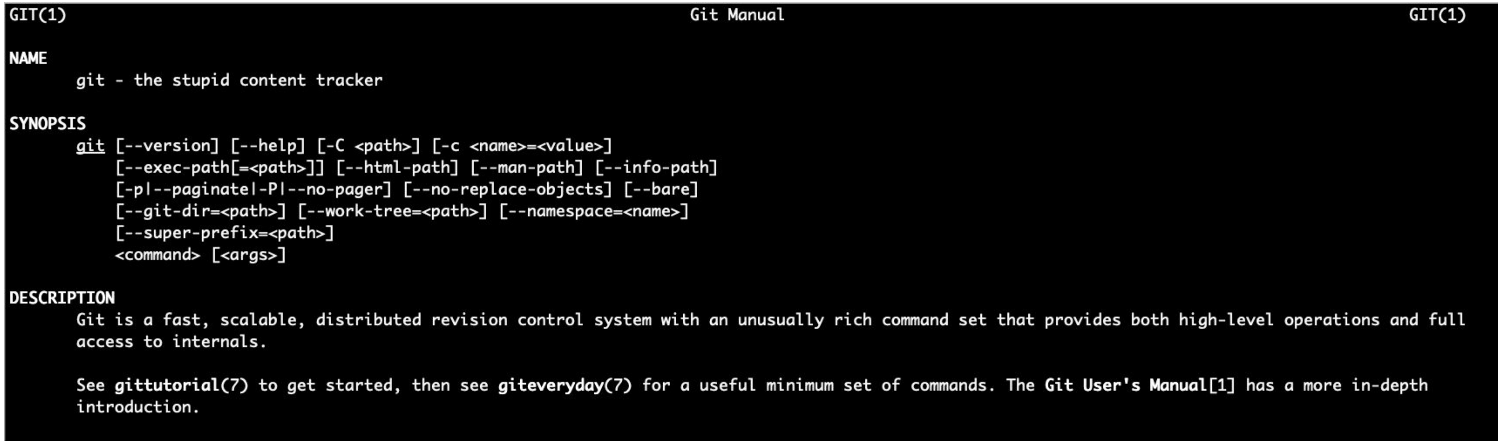
- Who made the change?
    - So you know whom to blame
- What has changed (added, removed, moved)?
    - Changes within a file
    - Addition, removal, or moving of files/directories
- Where is the change applied?
    - Not just which file, but which version or branch
- When was the change made?
    - Timestamp
- Why was the change made?
    - Commit messages



Tracking file changes

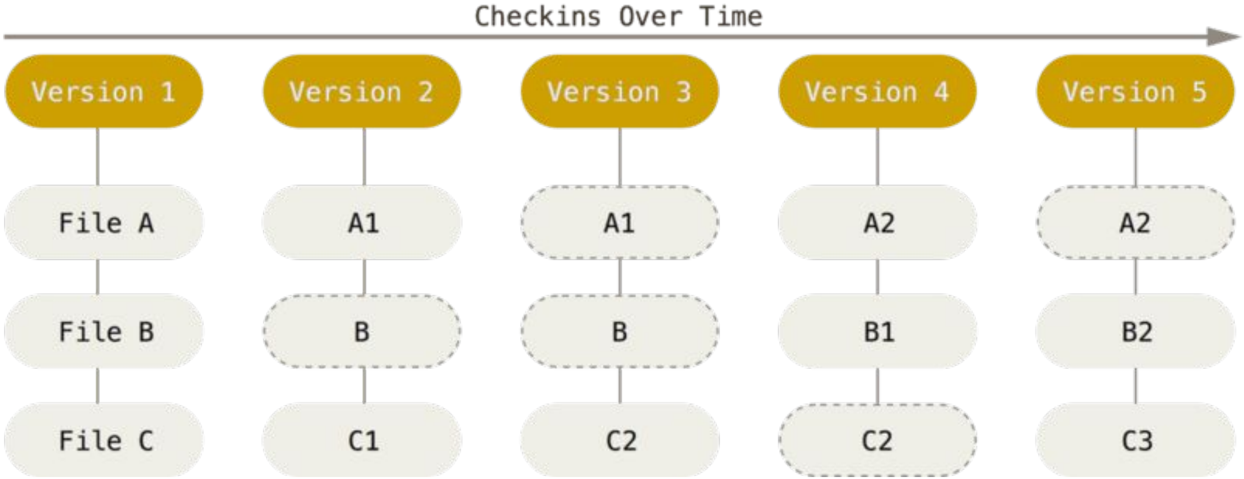
Git

- Started by Linus Torvalds – 2005
- Efficient for large projects
    - E.g. Linux
    - Much faster than other alternatives



```
GIT(1)                                  Git Manual                                  GIT(1)

NAME
       git - the stupid content tracker

SYNOPSIS
       git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|-P|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>]
           <command> [<args>]

DESCRIPTION
       Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full
       access to internals.

       See gittutorial(7) to get started, then see giteveryday(7) for a useful minimum set of commands. The Git User's Manual[1] has a more in-depth
       introduction.
```

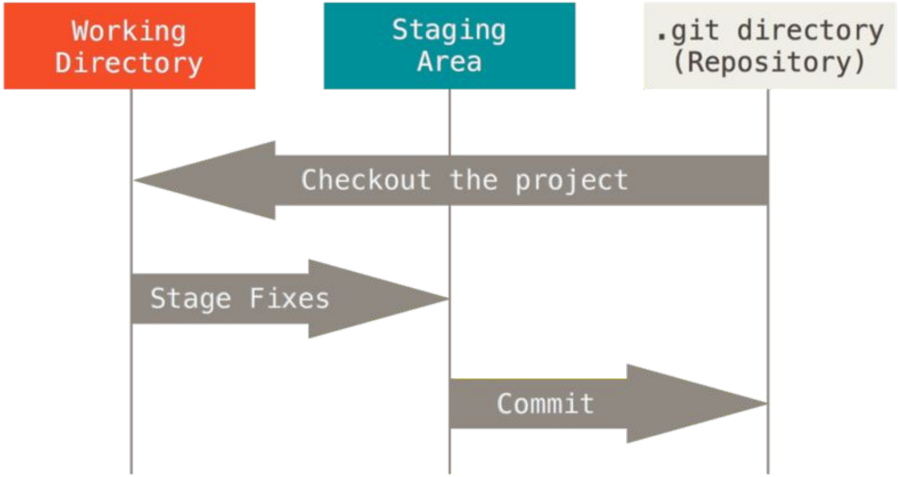
Git



Stream of Snapshots

Git

- **Modified**
  - You have changed the file but have not committed it to your database yet.
- **Staged**
  - You have marked a modified file in its current version to go into your next commit snapshot.
- **Committed**
  - the data is safely stored in your local database.

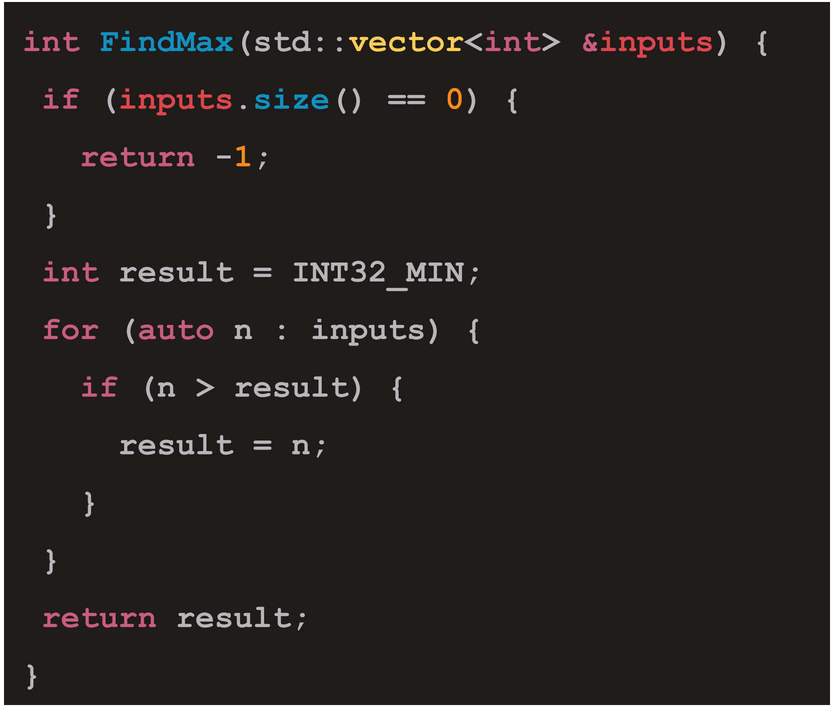


Main sections of a Git project

Runtime Analysis

- How long does it take for our algorithm to finish?
- Depends on both Algorithm and Input
  - Average case OR
  - Worst case

2	4	6	1	8



```
int FindMax(std::vector<int> &inputs) {
 if (inputs.size() == 0) {
   return -1;
 }
 int result = INT32_MIN;
 for (auto n : inputs) {
   if (n > result) {
     result = n;
   }
 }
 return result;
}
```

Finding which item is faster?

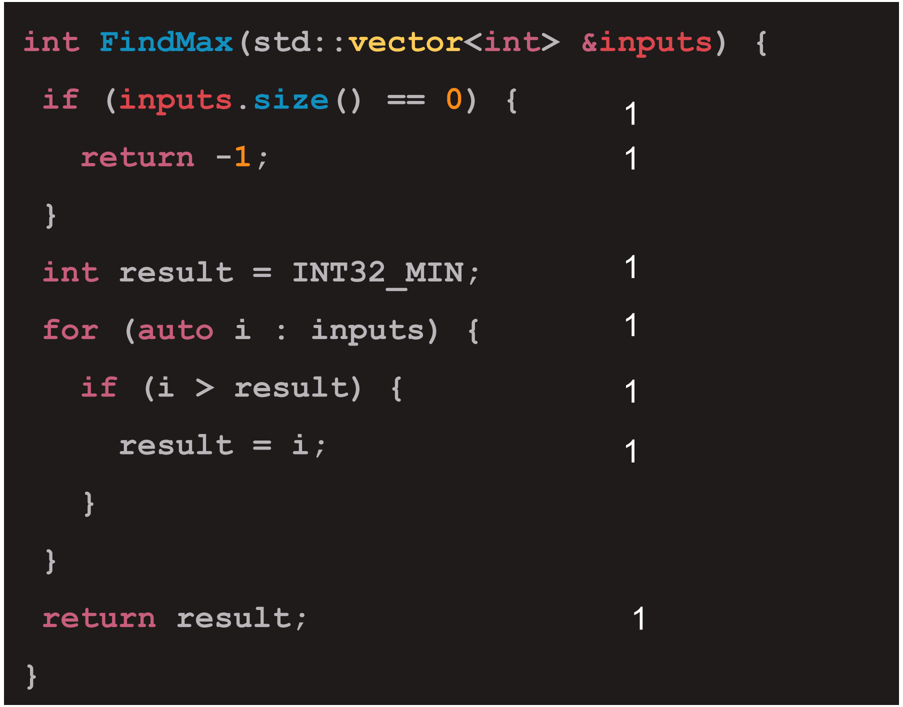
Algorithm A: starts from left Machine 1: 10ms, 100ms
Machine 2: 200ms, 10ms

Algorithm B: starts from right 100ms, 10ms

The runtime grows with the size of the array

Runtime Analysis

- It is hard to compare run times
  - Depends on the things like hardware, OS, …
- Count the number of operations
  - What is an operation?
    - i++
    - i = i / 2
    - I = i + 2
- For input *I* of size *n: R = F(I,n)*
- T(*n*)*:* worst case
  - Usually we care about the worst case input
- Basic operations:
  - Basic math operations +, -, *, /
  - Comparison
  - Assignment



```
int FindMax(std::vector<int> &inputs) {
 if (inputs.size() == 0) {             1
   return -1;                          1
 }
 int result = INT32_MIN;               1
 for (auto i : inputs) {               1
   if (i > result) {                   1
     result = i;                       1
   }
 }
 return result;                         1
}
```

- Number of operations:
  - If n=0: 2
  - If n>0: **1 + 1 + n(1 + 1 + k) + 1**
    - k is between 0 and **1**
    - **Worst case**: **3n + 3 < 4n < 9n < 10n < 100n^100**
      - **If n is close to 0, T(n) = 3**
      - **If n is too high, T(n) ≈ 3n**

…To be more precise:

When we do runtime analysis, we are really talking about the **number of operations** based on **the number of inputs**, which does affects the **runtime**.

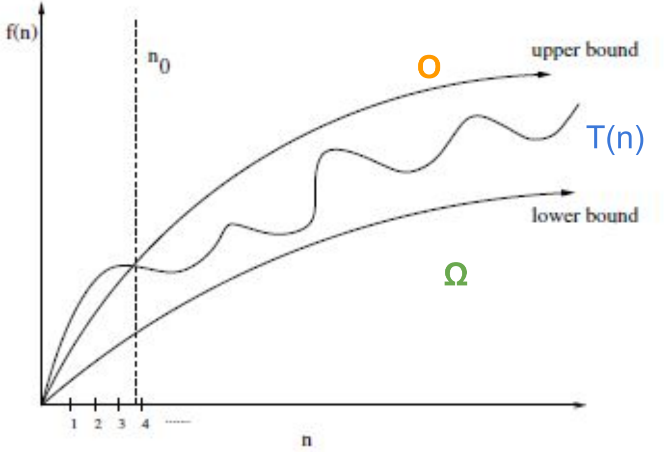
Runtime analysis = F(n), in the worst case, when n gets close to infinity.

Big O, Ω, ϴ

- Let **a** and $\mathbf{n_0}$ be constants
- T(n) is O(f(n)) if…
  - T(n) < **a***f(n) for some n > n_0
  - where a and n0 are constants
  - Essentially an upper-bound
- T(n) is said Ω(f(n)) if…
  - T(n) > **b***f(n) for some n > n_0
  - Essentially a lower-bound
- T(n) is said to be Θ(f(n)) if…
  - T(n) is both O(f(n)) AND Ω(f(n))

5n+2 = O(1) ? 5n+2 < a.1 , No!

5n+2 = Ω(1) ? 5n+2 > a.1 -> a=1 -> Yes!

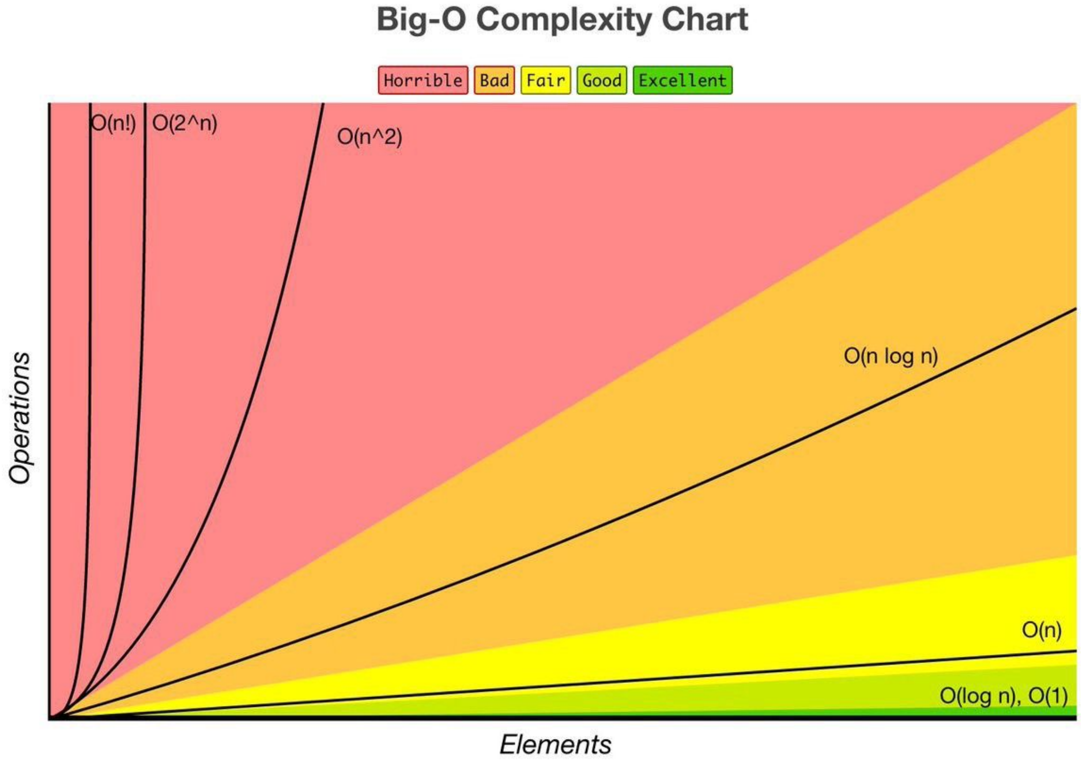


T2(n) = 5n + 2 = O(n) —> f(n)=n → 5n+2 < a.n → 5n+2 < 6n (when n is really high) → 4 < (5n+2)/n < 6
T2(n) = 5n + 2 = Ω(n) —> f(n)=n → 5n+2 > a.n → 5n+2 > 4n (when n is really high)→ 4 < (5n+2)/n < 6

5n+2 = O(n^2) = O(n^3)

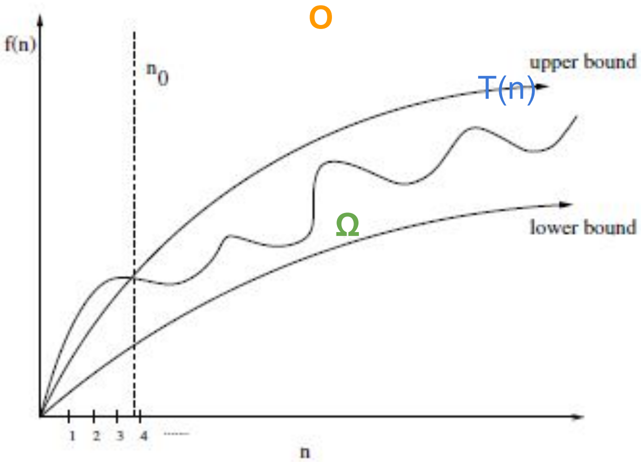
5n+2 = Θ(n) → tight bound
5n+2 != Ω(n^2)
5n+2 != Θ(n^2)
5n+2 = O(n) = O(n^2)= O(n^3) →-100 < **-10** < 1 < **10** < 100 < 1000

Big O, Ω, ϴ

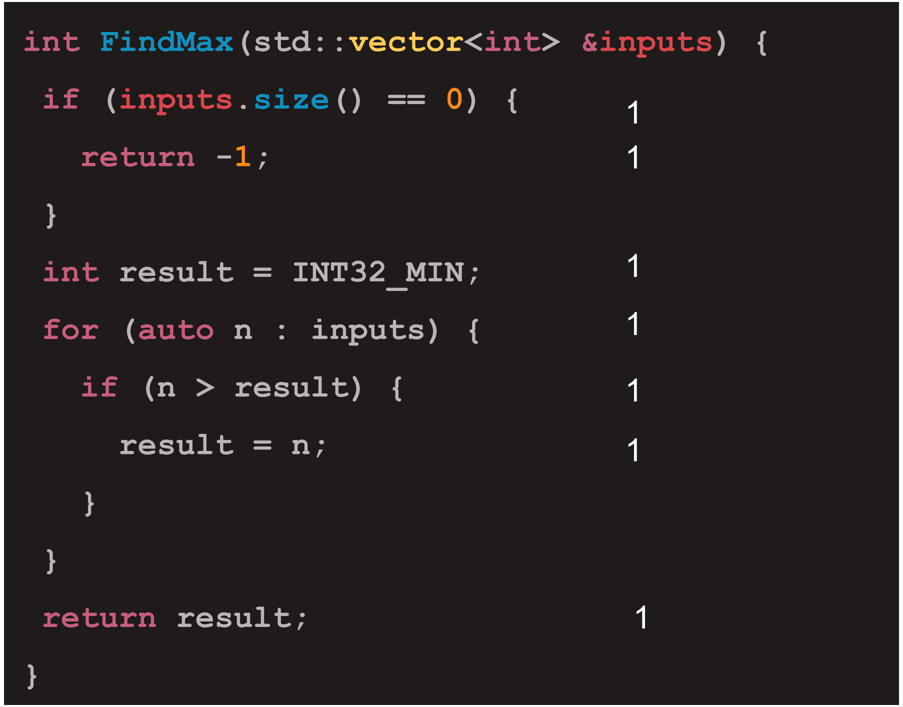

Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

Some examples

- T(n) = f(n) + c = Θ(f(n))
    - T(n) = 5n+10 = Θ(5n+10)= Θ(n)
- T(n) = c.f(n) = Θ(f(n))
    - T(n) = 5n+10 = Θ(5n+10)= Θ(n)
- T(n) = n^2 + n = Θ(n^2)
- T(n) = n^2 + log(n) = Θ(n^2)
- T(n) = n + nlog(n) = Θ (nlog(n))



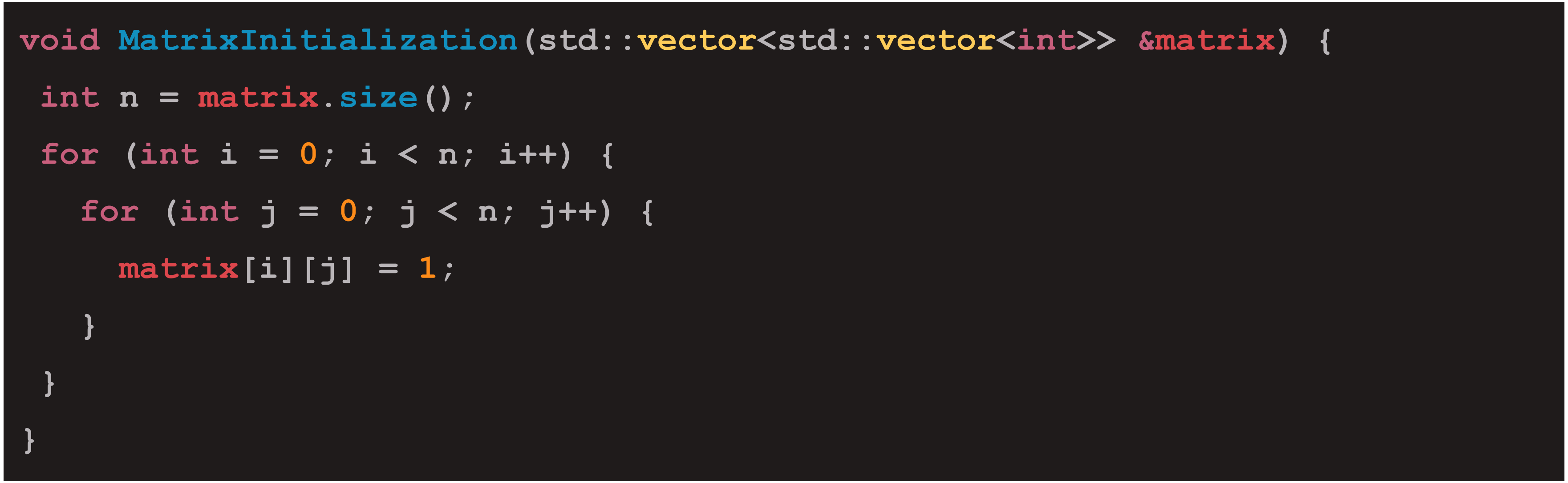
Runtime Analysis



```
int FindMax(std::vector<int> &inputs) {
 if (inputs.size() == 0) {          1
   return -1;                       1
 }
 int result = INT32_MIN;            1
 for (auto n : inputs) {            1
   if (n > result) {                1
     result = n;                    1
   }
 }
 return result;                      1
}
```

- Number of operations:
  - If $n=0$: 2
  - If $n>0$: $1 + 1 + 1 + n(1 + 1 + k) + 1$
    - k is between 0 and 1
    - Worst case: **$3n + 4$** = **$O(n)$**

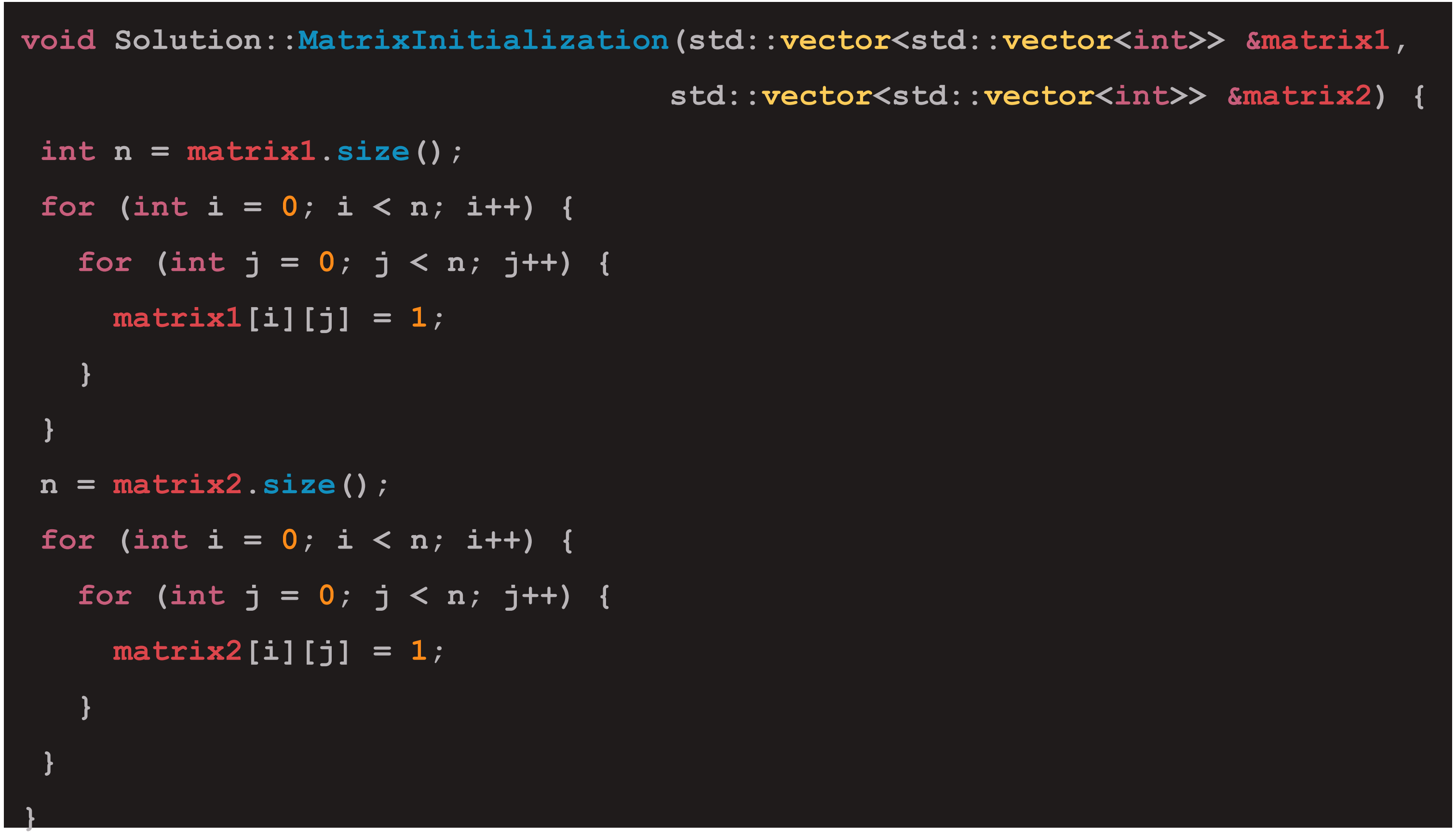
Runtime Analysis



```cpp
void MatrixInitialization(std::vector<std::vector<int>> &matrix) {
 int n = matrix.size();
 for (int i = 0; i < n; i++) {
   for (int j = 0; j < n; j++) {
     matrix[i][j] = 1;
   }
 }
}
```

T(n)= 1 + n (2 + n (2 + 1)) = 1 + n (2 + 3n) = 3n^2 + 2n + 1 = O(n^2)

$$\sum_{1}^{n}\sum_{1}^{n} 1 = \sum_{1}^{n} n = n \cdot n = n^2$$



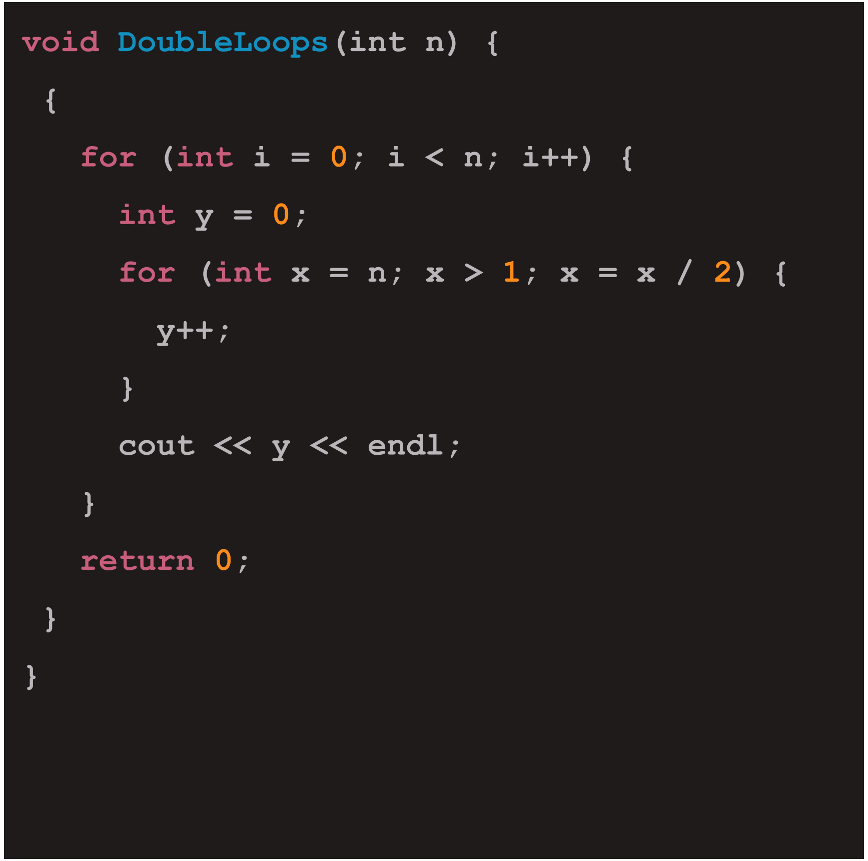
```
void Solution::MatrixInitialization(std::vector<std::vector<int>> &matrix1,
                                    std::vector<std::vector<int>> &matrix2) {
 int n = matrix1.size();
 for (int i = 0; i < n; i++) {
   for (int j = 0; j < n; j++) {
     matrix1[i][j] = 1;
   }
 }
 n = matrix2.size();
 for (int i = 0; i < n; i++) {
   for (int j = 0; j < n; j++) {
     matrix2[i][j] = 1;
   }
 }
}
```

T(n)= 1 + n (2 + n (2 + 1)) = 1 + n (2 + 3n) = 3n^2 + 2n + 1 = O(n^2)

new_T(n) = 2T(n) = O(n^2)

Runtime Analysis

- **Careful about nested loops**
    - We can’t alway look at the number of nested loops and raise n to that power!
- Carefully count the operations
    - Outer loop increments by 1 each time
    - Inner loop updates by dividing x in half each iteration
        - After 1st iteration => x=n/2
        - After 2nd iteration => x=n/4
        - After 3rd iteration => x=n/8
        - After the last iteration (k)
            - x = n/2^k = 1.
            - Solve for k: k = log2(n) iterations
    - O(n*log(n))



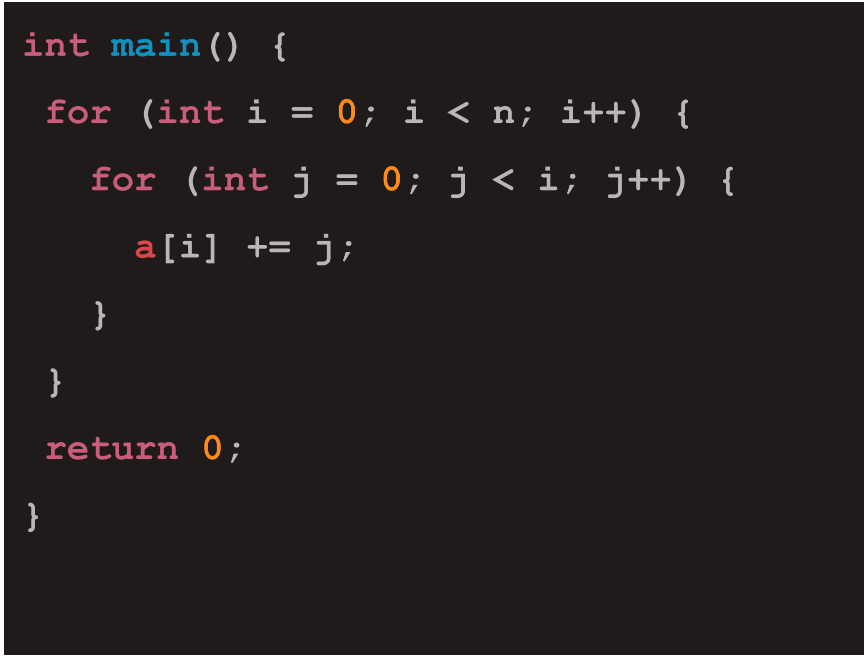
```
void DoubleLoops(int n) {
 {
   for (int i = 0; i < n; i++) {
     int y = 0;
     for (int x = n; x > 1; x = x / 2) {
       y++;
     }
     cout << y << endl;
   }
   return 0;
 }
}
```

Some Base Sums

- Arithmetic Series $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$
- Geometric Series $\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1} = \Theta(c^n)$
- Harmonic Series $\sum_{i=1}^{n} 1/i = \Theta(\log n)$

Runtime Analysis

- What is the runtime of this function?



```
int main() {
 for (int i = 0; i < n; i++) {
   for (int j = 0; j < i; j++) {
     a[i] += j;
   }
 }
 return 0;
}
```

$$\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} i = ?$$

What about recursion?

- What is the runtime of this function?

$T(n) = 1 + T(n-1)$

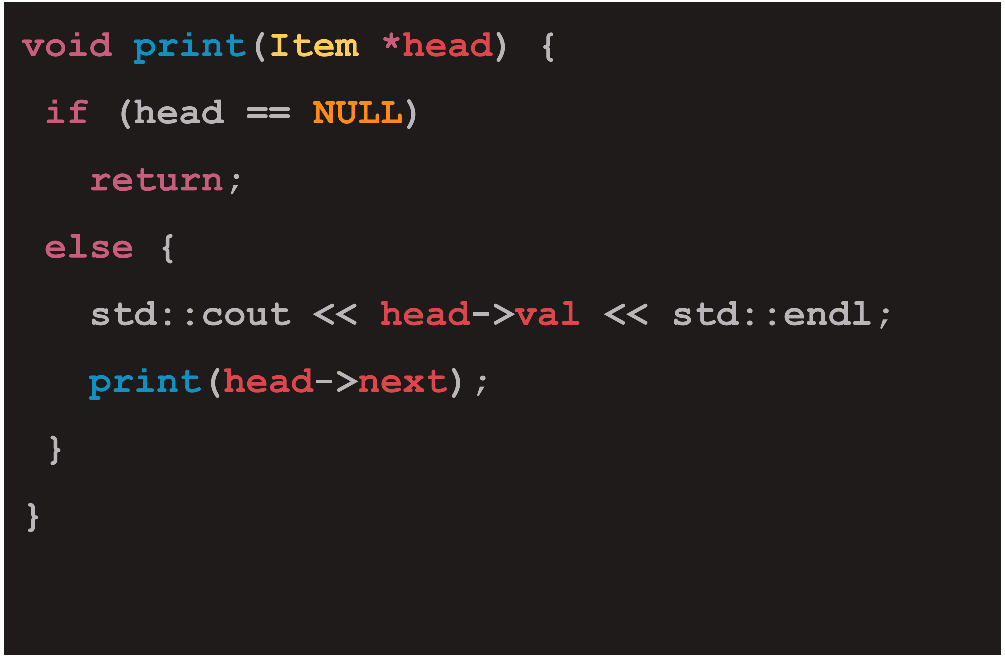
$= 1 + 1 + T(n-2)$

$= 1 + 1 + 1 + T(n-3)$

$= 1 + 1 + 1 + \ldots + 1$

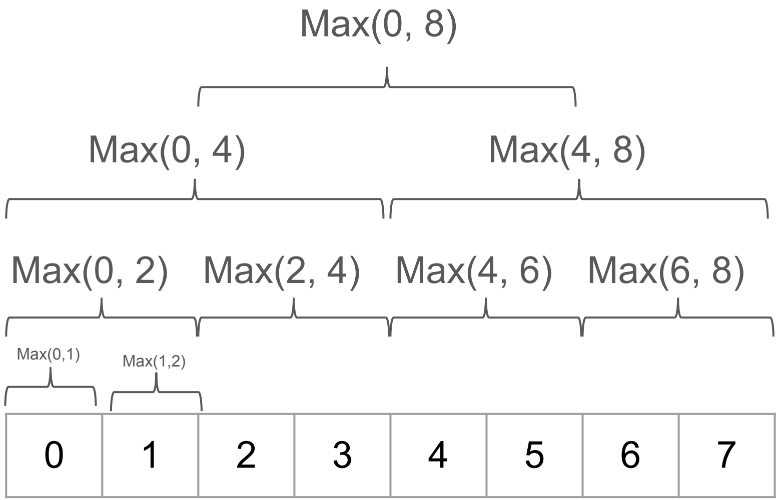
$= O(n)$

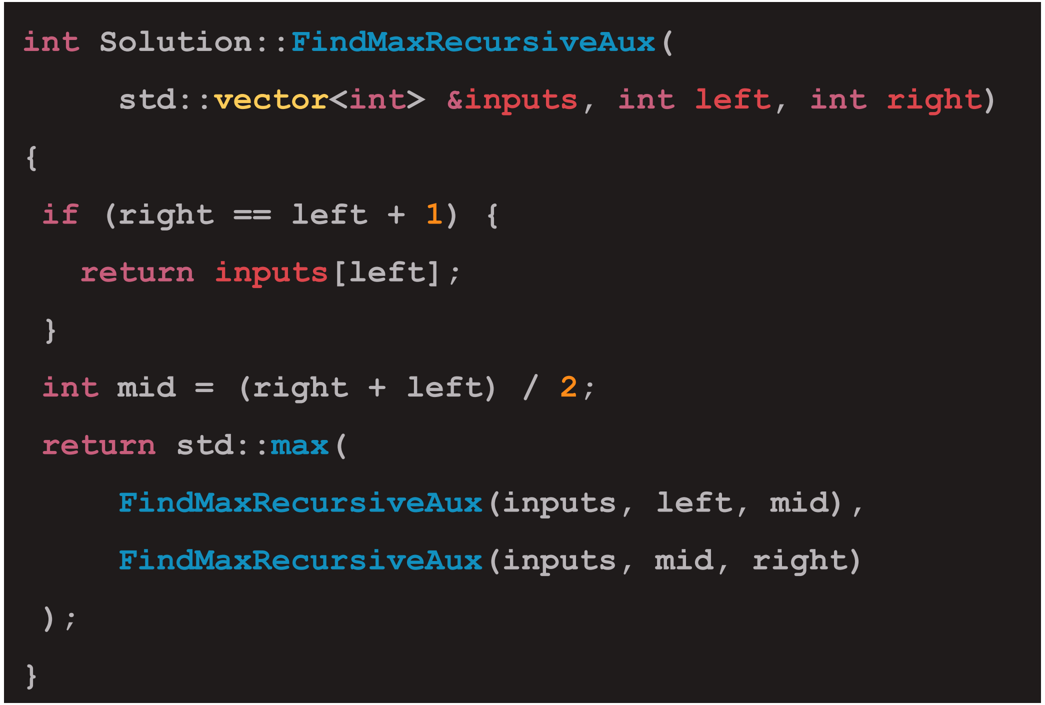
(1, 2, 4, 5, 6)
1 → (2, 4, 5, 6)
2 → (4, 5, 6)
4 → (5, 6)
5 → 6



```
void print(Item *head) {
 if (head == NULL)
   return;
 else {
   std::cout << head->val << std::endl;
   print(head->next);
 }
}
```

Recursive FindMax





```cpp
int Solution::FindMaxRecursiveAux(
     std::vector<int> &inputs, int left, int right)
{
 if (right == left + 1) {
   return inputs[left];
 }
 int mid = (right + left) / 2;
 return std::max(
     FindMaxRecursiveAux(inputs, left, mid),
     FindMaxRecursiveAux(inputs, mid, right)
 );
}
```

- We visit every element only once, so T(n) should be O(n)
- T(n)= 2T(n/2) + Θ(1) = O(n)
    - We don’t provide the proof now