EE538: Computing Principles for Electrical Engineers

Discussion 2: Time Complexity & C++ Basic

University of Southern California
05/25,05/26 Summer 2022
Instructor: Arash Saifhashemi
TA and Mentors: Yijun Liu, Aditi Bodhankar, Zixu Wang

Logistics

- Homework 1 submission
  - The latest submission will be graded.
- Late policy
  - For all homework in the future, you may have up to 3 late days per homework.
    - Suppose t is the deadline, a submission in (t, t+24h] is consider to be one day late.
    - 1 day late is -15, 2 day late is -30 points, 3 day late is -45 points
    - More than 3 days late is a 0
- Grading timeline
  - Grading may take up to two weeks after the due date.
  - You can find your grades in blackboard.
- Dispute of grades
  - 2 weeks after the grades being released.

Homework Submissions Demo

- Homework 1 Piazza post
- Deadline: Tuesday 06/03/2022 12:00

note @19

Homework 1 released

Hi all,

Homework 1 is out. It is due next Friday noon (June 3rd, 2022, 12pm).

We use Github classroom for homeworks. You will need to submit your homework through Github with git, and your homework will be graded based on your submission on Github.

Here is the invitation link to HW 1 on Github: https://classroom.github.com/a/BAVV-ic0

Note that we check the time of your last commit as your final submission time. Please make sure you do not push after the deadline --- it will cause a late penalty!

Your repo will be created!

You're ready to go!

You accepted the assignment, **HW1**.

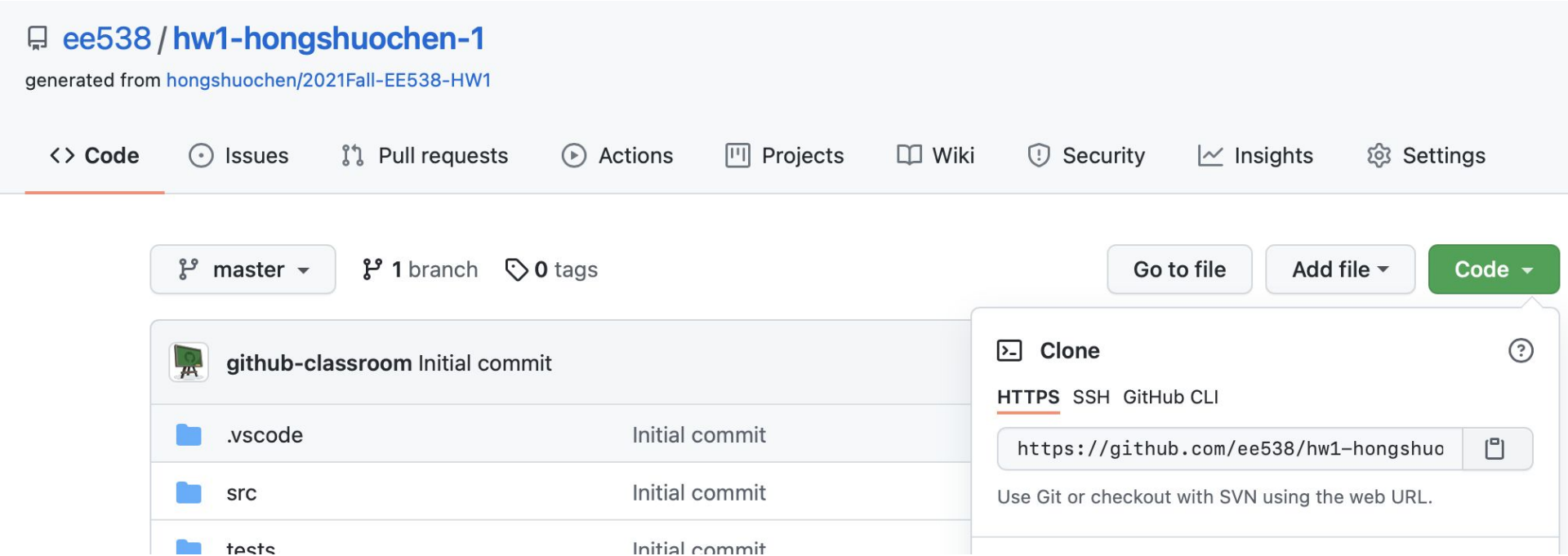
Your assignment repository has been created:

`https://github.com/ee538/hw1-hongshuochen-1`

We've configured the repository associated with this assignment (update).

Note: You may receive an email invitation to join ee538 on your behalf. No further action is necessary.

Clone the repo and start working



Read ReadME carefully to fulfill the requirements

HW1 EE538 - Computing Principles for Electrical Engineers

- Please clone the repository, edit README.md to answer the questions, and fill up functions to finish the homework.
- For non-coding questions, fill out the answers below each question. Please write your answer there.
- For coding questions, please make sure that your code can run `bazel run/test`. In this homework, you will need to fill up cpplib.cc and tests in tests. **Do Not change or modify any given function names and input or output formats in both cpplib.cc and tests in tests. Unexpected changes will result in zero credit.**
- For coding questions, there is a black box test for each question. All points are only based on passing the test cases or not (i.e. we don't grade your work by your source code). So, try to do comprehensive testing before your final submission.
- For submission, please push your answers to Github before the deadline.
- Deadline: **Tuesday, September 7th by 6:30 pm**

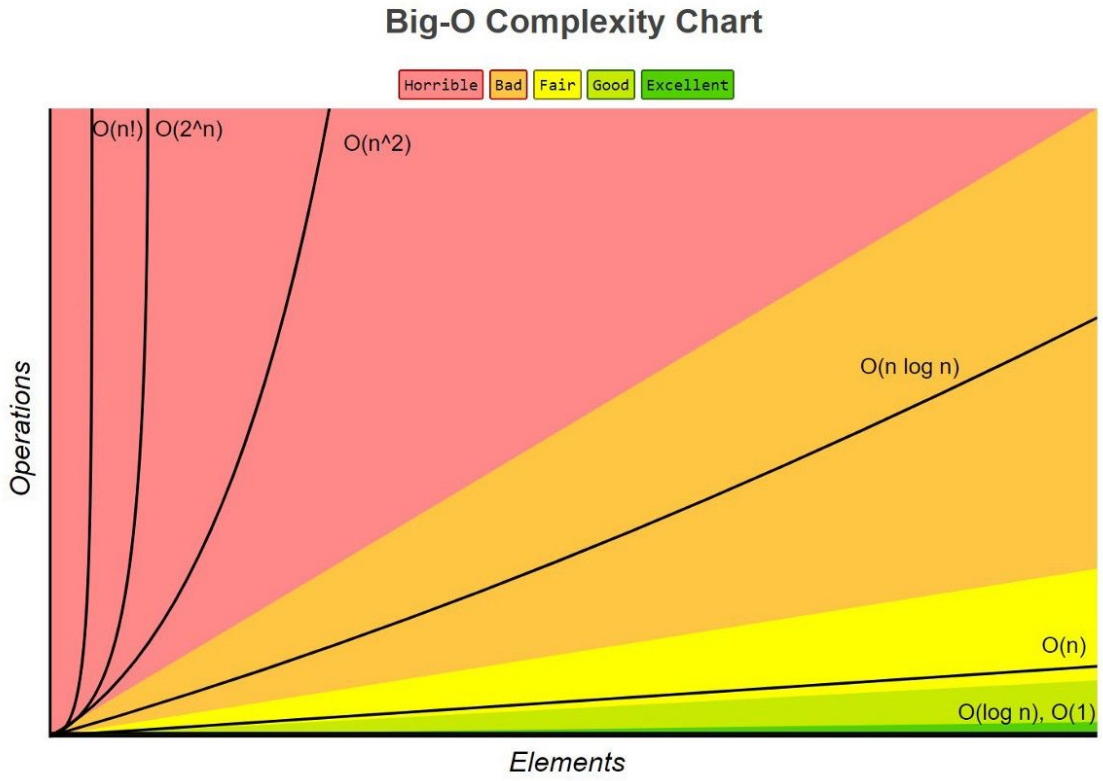
Modify the repo in your local computer

- Submission
  - git add .
  - git commit -m “messages”
  - git push
- You could push multiple times, we only grade your last submission!
- **Don’t push after deadlines or you will get some penalties!**

Outline

- Time Complexity Analysis
  - Loop based structure
- C++ practice
  - Basic coding examples
  - Unit tests
  - Recursion examples
  - String manipulation

Time complexity chart



Loop Based Structure - Easy example

- Matrix Multiplication
  - Assume the size of input matrix is n x n.
  - Time complexity?

$$\mathbf{C} = \mathbf{AB}$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix} \quad \mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

For each element in output matrix C, we need to do **n** multiplications and **n-1** additions.

There are **n^2** elements, so the total number of operations is **n^2 x (2*n-1) -> $O(n^3)$**

Loop Based Structure - Easy example

```
// Assume the input has two n by n matrices A and B (n > 0)
// The output is also an n by n matrix.
vector<vector<int>> Multiply(const vector<vector<int>>& A, const vector<vector<int>>& B) {
    int n = A.size();
    vector<vector<int>> C(n, vector<int>(n, 0));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++ j) {
            C[i][j] = 0;
            for (int k = 0; k < n; ++k) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}
```

Constant operations

Loop Based Structure - Easy Example

- What is the time complexity of below function?
  - Hint: how many iterations will this while loop run?

```
void Example1(int n) {
    int i = 1;
    while (i < n) {
        i *= 2;
    }
}
```

Solution

- The loop will run ~log(n) iterations.
- In i^{th} iteration, the number of operations is constant!

- Time complexity = O(logn)

Example:

- 1, 2, 4, 8, …, 2^i
- 2^0, 2^1, 2^2, 2^3, …, 2^i => i = log n

Loop Based Structure - Hard Example

- Find all prime numbers that are not greater than n.
- Approach 1: Brute force
  - Let's suppose isPrime runs at O(sqrt(n)) time - Why?
  - What is the time complexity of this function?

```
vector<int> FindAllPrimeNumbers(int n) {
    vector<int> result;
    for (int i = 2; i <= n; ++i) {
        if (isPrime(i)) {
            result.push_back(i);
        }
    }
    return result;
}
```

Solution

- The loop will run ~n iterations.
- In i^{th} iteration, the number of operations is O(sqrt(n))!

- Time complexity = $\sum_{i=2}^{n} \sqrt{i} \rightarrow O(n\sqrt{n})$

Example:

- IsPrime(4): check if 2 || 4
- IsPrime(6): check if 2 || 6, no need to check 3 || 6 or above because: 3 > sqrt(6), m = 6/3 = 2 < 3 has been checked.

Loop Based Structure - Hard Example

- Find all prime numbers that are not greater than n.
- Approach 1: (modified) Sieve of Eratosthenes

```
vector<int> FindAllPrimeNumbersSieve(int n) {
    vector<int> result;
    vector<bool> visited(n + 1, false);
    for (int i = 2; i <= n; ++i) {
        if (!visited[i]) {
            result.push_back(i);
            for (int j = i + i; j <= n; j += i) {
                visited[i] = true;
            }
        }
    }
    return result;
}
```

Example:

- 2 is a prime number.
- Then, any number that's multiple of 2 is not a prime number.
- Mark those numbers as visited so that you can skip them in future iterations.

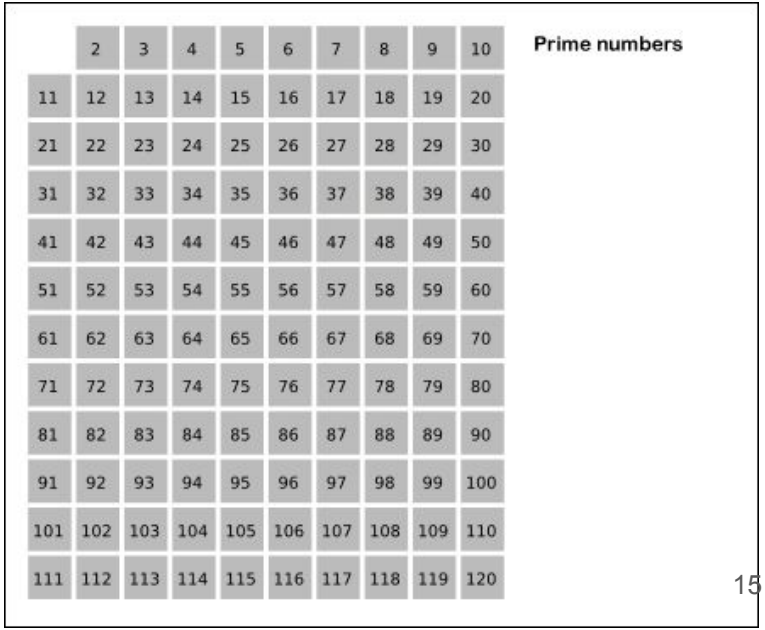
- 3 is a prime number.
- Repeat…

- Starting from 2, any number that's not been visited is a prime number.

Loop Based Structure - Hard Example

- Find all prime numbers that are not greater than n.
- Approach 1: (modified) Sieve of Eratosthenes
  - What is the time complexity of this function?

```cpp
vector<int> FindAllPrimeNumbersSieve(int n) {
    vector<int> result;
    vector<bool> visited(n + 1, false);
    for (int i = 2; i <= n; ++i) {
        if (!visited[i]) {
            result.push_back(i);
            for (int j = i + i; j <= n; j += i) {
                visited[i] = true;
            }
        }
    }
    return result;
}
```



Loop Based Structure - Hard Example

- Find all prime numbers that are not greater than n.
- Approach 1: (modified) Sieve of Eratosthenes
  - What is the time complexity of this function?

```
vector<int> FindAllPrimeNumbersSieve(int n) {
    vector<int> result;
    vector<bool> visited(n + 1, false);
    for (int i = 2; i <= n; ++i) {
        if (!visited[i]) {
            result.push_back(i);
            for (int j = i + i; j <= n; j += i) {
                visited[i] = true;
            }
        }
    }
    return result;
}
```

Solution

- Harmonic series
  $1 + \frac{1}{2} + \frac{1}{3} + \ldots + 1/n \sim \log n$
- Time complexity = $O(n(1 + \frac{1}{2} + \frac{1}{3} + \ldots + 1/n)) = O(n\log n)$
- $T(n) = O(n + n/2 + n/3 + n/5 + n/7\ldots) = O(n\log(\log n))$
- Proof

Some Useful Series

- 1 + ½ + ⅓ + … + 1/n = O(logn)
- 1 + ¼ + 1/9 + 1/16 + … + 1/n^2 = O(1)
  - The sum will converge to pi^2/6, also called Basel Problem.

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}.$$

- 1 + 2 + 3 + … + n = n(n+1)/2 = O(n^2)
- 1 + 4 + 9 + 16 + … + n^2 = n(n+1)(2n+1)/6 = O(n^3)
- 1 + 2 + 4 + 8 + 16 + 32 + … + 2^n = 2^(n+1)-1 = O(2^n)

Outline

- Time Complexity Analysis
  - Loop based structure
- C++ practice
  - Basic coding examples
  - Unit tests
  - Recursion examples
  - String manipulation

Class

- C++ is an object-oriented programming (OOP) language
- Attributes: variables
- Methods: functions

For example:

1. Class: Car
2. Object: Tesla Model 3
3. Attributes: speed weight color
4. Methods: drive break stop

Class / Struct

- Only one difference:
  - The members in a **struct** are **public** by default.
  - The members in a **class** are **private** by default.
- Always use class in practice unless specified.

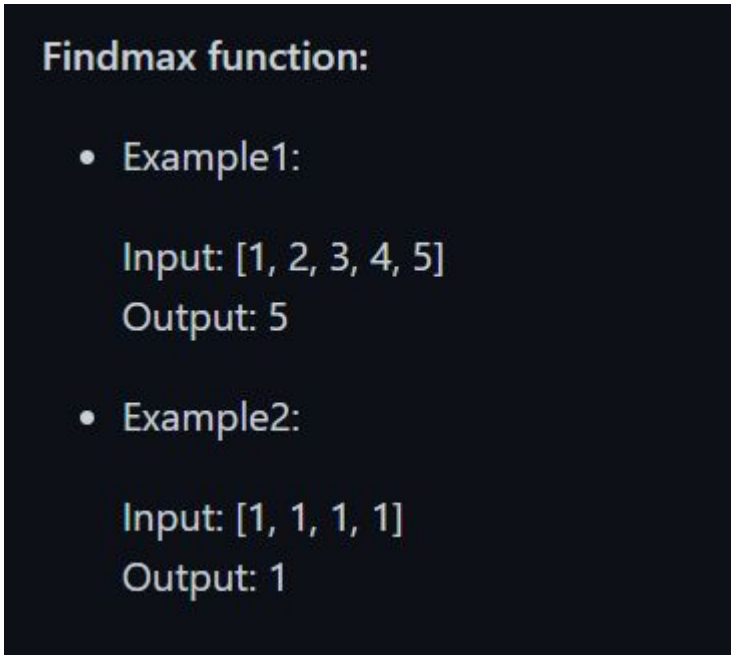
C++ Practice

- **Basic coding examples**
- **Unit tests**
- **Recursion examples**
- **String manipulation**

Clone the following repo first

git clone https://github.com/Zi-xu-Wang/Discussion2_Demo.git

Question 1: Find the maximum value in a vector



Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q1_student_test*** to run the tests of this function

Sample Solution

```
int CPPLib::Findmax(std::vector<int> &input){
    if(input.size() == 0){return -1;}
    int max = -INT_MAX;
    for(int i=0; i< input.size();i++){
        if (input[i] > max){
            max = input[i];
        }
    }
    return max;
}
```

Use: ***bazel run src/main:main*** to call this function in the main function

C++ Practice

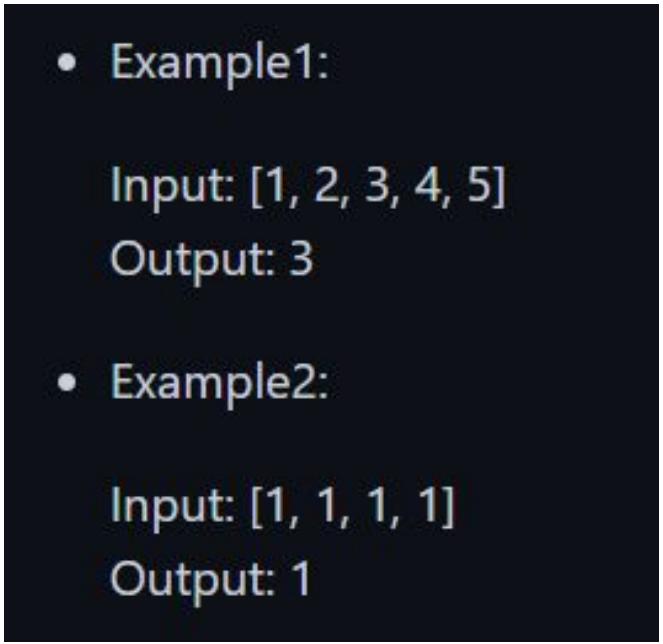
- **Basic coding examples**
- **Unit tests**
- **Recursion examples**
- **String manipulation**

Sample Test

```
// Add your own tests in this file
TEST(Q1_Student, Findmax_Test_1) {
    CPPLib s;
    std::vector<int> input = {1,2,3,4,5};
    int exp = 5;
    int act = s.Findmax(input);
    EXPECT_EQ(exp, act);
}
```

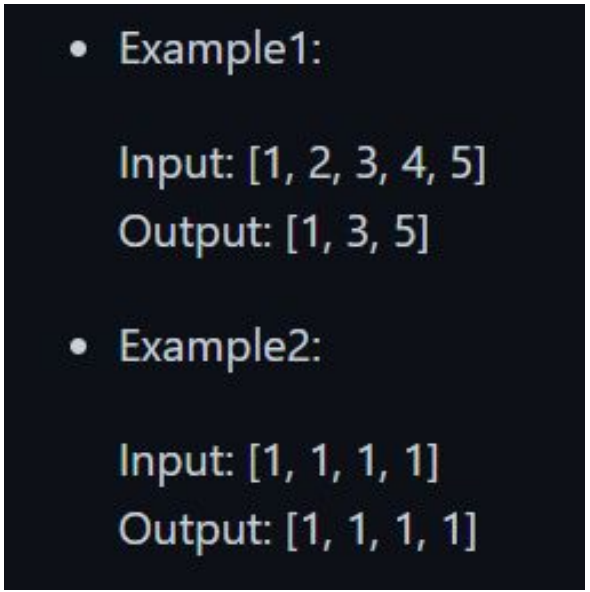
Use: ***bazel test src/tests:q1_student_test*** to run the tests of this function

Question 2: Find the average value of a vector



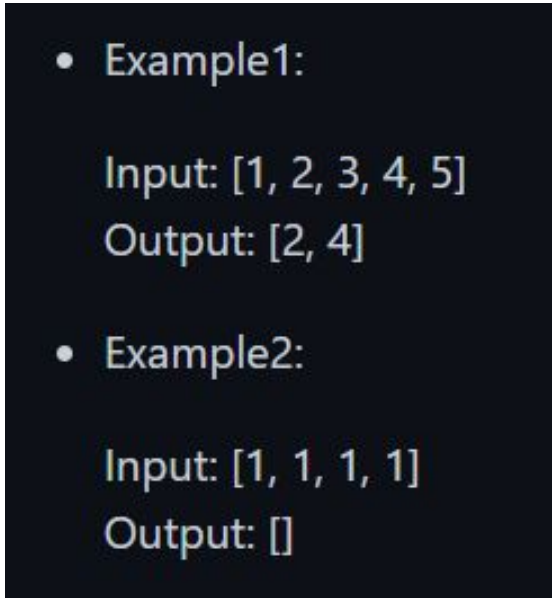
Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q2_student_test*** to run the tests of this function

Question 3: Find the odd numbers in a vector



Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q3_student_test*** to run the tests of this function

Question 4: Find the even numbers in a vector

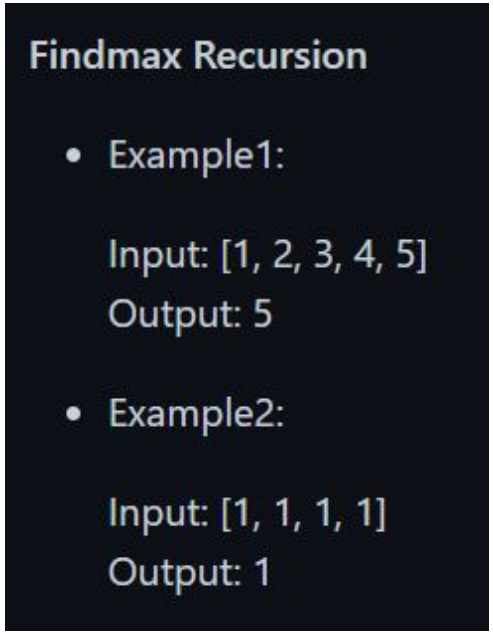


Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q4_student_test*** to run the tests of this function

C++ Practice

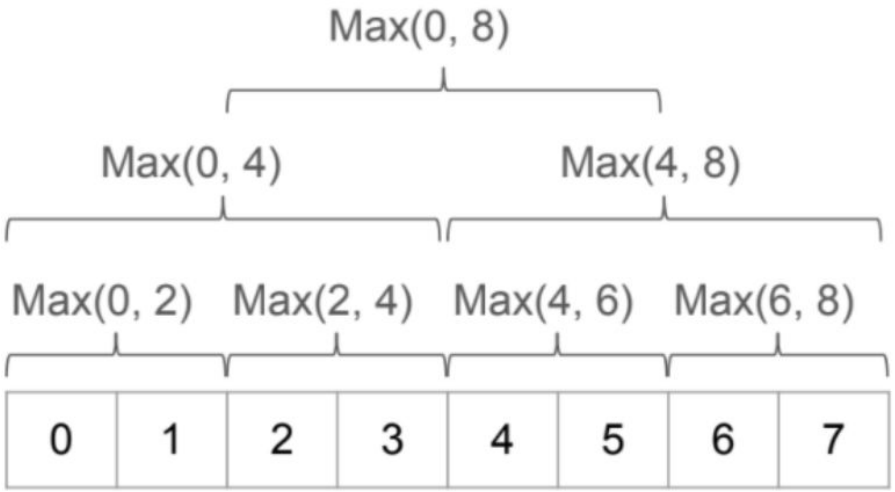
- **Basic coding examples**
- **Unit tests**
- **Recursion examples**
- **String manipulation**

Question 5: Recursive Findmax



Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q5_student_test*** to run the tests of this function

Hint:

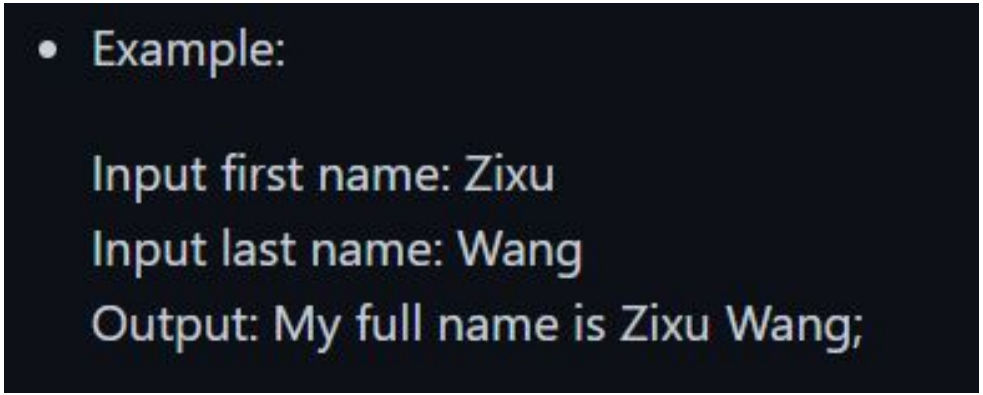


Sample solution:

```
// Q5
int CPPLib::FindmaxR(std::vector<int> &input, int left, int right){
    if(right == left+1){return input[left];}
    int mid = (left + right ) / 2;
    return std::max(FindmaxR(input, left, mid), FindmaxR(input, mid, right));
  }
```

Question 6: String Manipulation

Part a): Print Introduction



Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q6_student_test*** to run the tests of this function

Question 6: String Manipulation

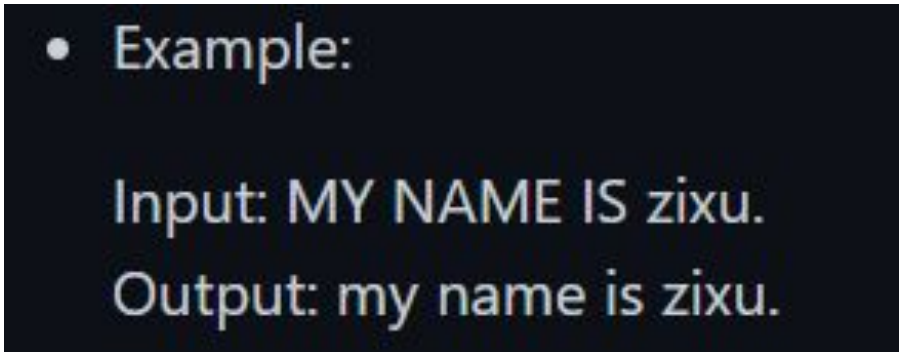
Part a): Print Introduction

```
std::string first;
std::string last;
std::cout<<"Input first name: "<<std::endl;
std::cin>>first;
std::cout<<"Input last name: "<<std::endl;
std::cin>>last;
std::string fullname = first + " " + last;
std::cout<< "My full name is: "<<fullname<<std::endl;
```

Use: ***bazel run src/main:main*** to call this function in the main function
Use: ***bazel test src/tests:q6_student_test*** to run the tests of this function

Question 6: String Manipulation

Part b): To lower case



Question 6: String Manipulation

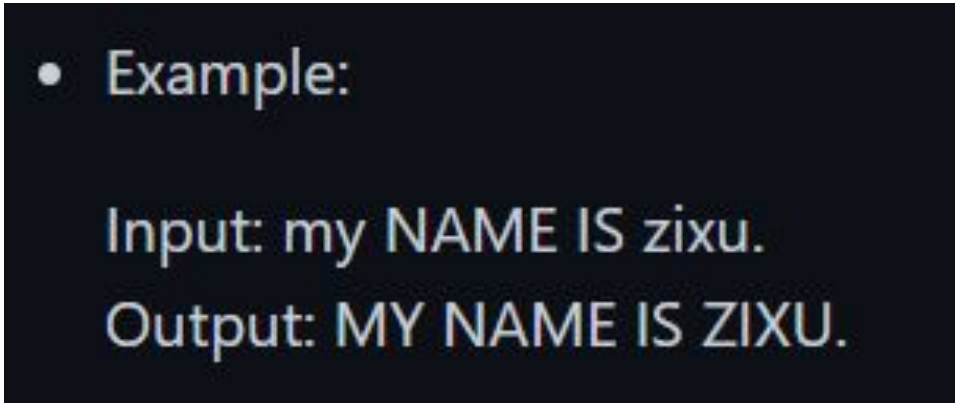
Part b): Hint:

```c
/* tolower example */
#include <stdio.h>
#include <ctype.h>
int main ()
{
  int i=0;
  char str[]="Test String.\n";
  char c;
  while (str[i])
  {
    c=str[i];
    putchar (tolower(c));
    i++;
  }
  return 0;
}
```

https://www.cplusplus.com/reference/cctype/tolower/

Question 6: String Manipulation

Part c): To upper case



Question 6: String Manipulation

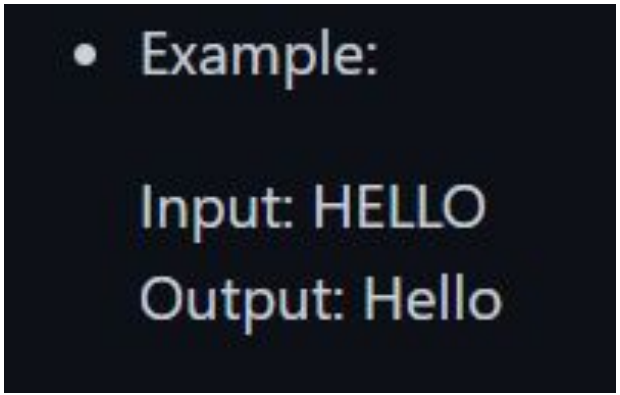
Part c): Hint:

```c
/* toupper example */
#include <stdio.h>
#include <ctype.h>
int main ()
{
  int i=0;
  char str[]="Test String.\n";
  char c;
  while (str[i])
  {
    c=str[i];
    putchar (toupper(c));
    i++;
  }
  return 0;
}
```

https://www.cplusplus.com/reference/cctype/toupper/

Question 6: String Manipulation

Part d): To title case



Question 6: String Manipulation

Part e): To random case