CS4414 Recitation 1

Course Introduction and C++ Setup

08/27/2021

Sagar Jha, Alicia Yang

About TA --- Sagar

- Senior PhD student in CS
- Advised by Prof. Birman on distributed systems (with a focus on RDMA networks)
- TA experience at Cornell
  - Practicum in Database Systems (Fall ‘16)
  - Cloud Computing (Spring ‘18, Spring ‘20)
  - Systems Programming (Fall ‘20, **Fall ‘21**)
- Office Hours
  - Thursday and Friday 5-7 pm (starting next week)
  - At https://cornell.zoom.us/j/99522656755?pwd=WTdzV1hFSzVIM1BLR0Q3TDZsRHdKdz09

Goals

Develop **systems** in **C++** that **perform well**

Goals

Develop **systems** in **C++** that **perform well**

For the recitation:

- Basic C++ proficiency: Read, write, and debug C++ code
- Working knowledge of Linux: The Linux command line and the filesystem

Goals for the recitation

- Basic C++ proficiency: Read, write, and debug C++ code
  - Standard containers – std::vector<T>, std::map<K, V>
  - pointers, iterators, templates, classes…
  - gdb for debugging, gprof for profiling
  - multi-threading, synchronization
- Working knowledge of Linux: The Linux command line and the filesystem

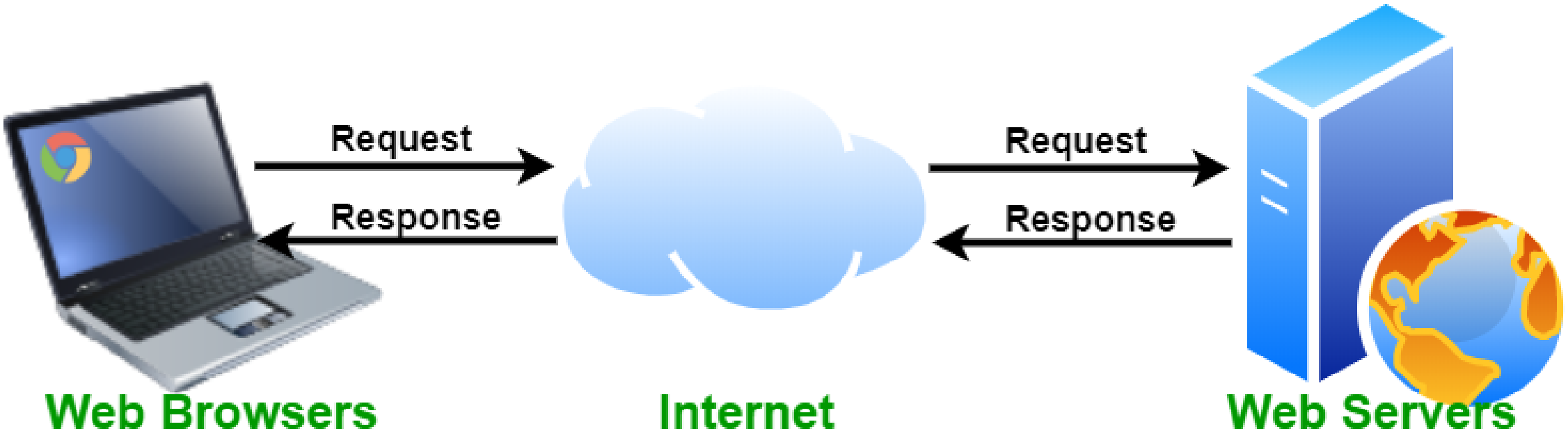
Secondary goals

- Learn to characterize code performance
- Make efficient use of hardware – learn to exploit CPU cores with threads
- Understand solutions to assignments/exams

System Performance will be a mainstay of this course!

What do we mean by performance?

- Latency: Time taken to compute
- Throughput: Number of operations per second


Request
Response
Request
Response
Web Browsers
Internet
Web Servers

Focus on system performance

It’s not just about

algorithm complexity. Why?

Reasoning about system performance

- Theoretical improvements don't always translate to better application runtimes

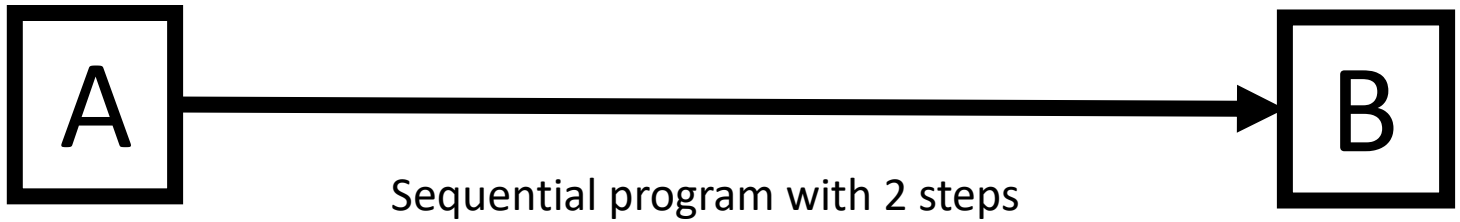
Insertion sort outperforms quick sort in some cases

Why?

1. Insertion sort is iterative – no overhead from recursive function calls (good for sorting a small set)
2. Insertion sort is fast when data is nearly sorted

Reasoning about system performance

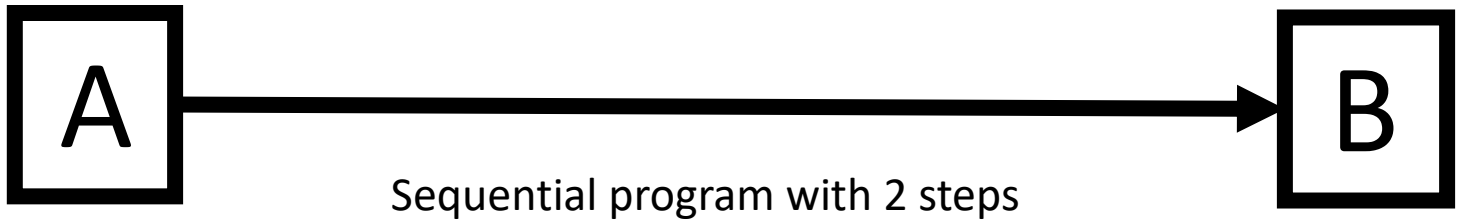
- Theoretical improvements don't always translate to better application runtimes
- Which algorithm? A system can be very complex with many features



- A = processing files, B = printing 1 million lines of output

Reasoning about system performance

- Theoretical improvements don't always translate to better application runtimes
- Which algorithm? A system can be very complex with many features



What if step A takes about 99% of the total time? We need to profile and understand performance characteristics of code we write!

Reasoning about system performance

- Theoretical improvements don’t always translate to better application runtimes
- Which algorithm? A system can be very complex with many features
- What if the code that implements the algorithm is inefficient?
- Sometimes heuristics work better

C++ Environment Setup

TA and Office Hours

Name	Offfice Hour 1	Office Hour 2	Zoom Link
Alicia Yang	Saturday 5-7PM	Sunday 6-7PM	https://cornell.zoom.us/j/93560684279?pwd=S1c0NjF5Y1ZLNnpVVU0xQll5K2tRUT09
Andrew D.	Tuesday 9-11AM	Thursday 10-11AM	https://cornell.zoom.us/j/91466505032?pwd=U01wTEQvTEx0T3NWcDdMeWMrdW1Zdz09
Zheng Wang	Monday 1-3PM	Wednesday 10-11AM	https://cornell.zoom.us/j/8812491232?pwd=VUtRWndqR2lvMjU1S1VZVkY5VkRxdz09
Sagar Jha	Thursday 5-7PM	Friday 5-7PM	https://cornell.zoom.us/j/99522656755?pwd=WTdzV1hFSzVIM1BLR0Q3TDZsRHdKdz09
Aahli Awatramani	Wednesday 9-10AM	Wednesday 2-4PM	https://cornell.zoom.us/j/92630999231?pwd=b0RkeVQ5TWczcWd1WGdLbXNpT21MQT09
Arthur Tanjaya	Tuesday 5-7PM	Thursday 5-6PM	https://cornell.zoom.us/j/3877784348?pwd=dkVwcFBwS1RDSHh2SXBhRXZVaVdtdz09
Muhammad Moughal	Monday 6-7PM	Tuesday 4-6PM	https://cornell.zoom.us/j/4905170673?pwd=V3dXS00wbEFleC9YSDBGS3Z4UTR5Zz09

About TA --- Alicia

- 1st year PhD student in CS, TAed this course Fall 2020
- Working with Prof. Birman in the area of distributed system
- Interested in scheduling and cluster management in machine learning system
- Office Hours **: Thursdays 6PM – 7PM**, **Saturdays 6PM - 7PM**
- Meeting by appointment for questions or assignment discussion

C++ Coding Environment

- C++ 20
- gcc-8 or recent
  - To check your gcc compiler version: $ g++ -v

C++ Coding Environment

- Editing Tools:
  - Visual Studio Code (link)
  - Emacs
  - Vi, …
- Compilation Tools: GNU Compiler Collection (GCC)
  - Cornell Engineering linux server remote access
  - Virtual Box
  - Most linux distributions have GCC
  - MacOS user has Clang compiler (not recommended for this course, since Clang and GCC are two different compiler and sometimes have different compilation results. The submitted assignments are run via GCC)

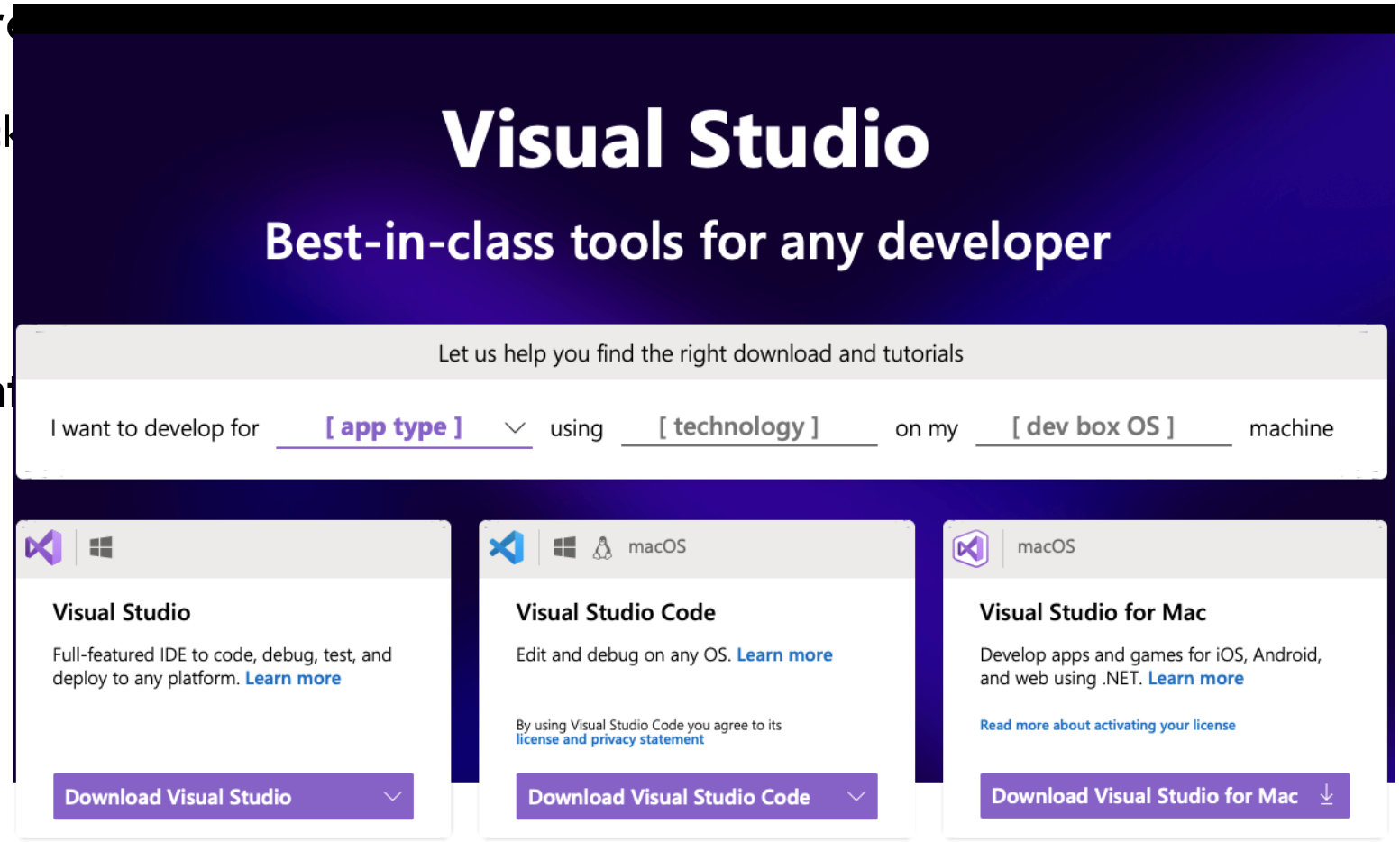
C++ Coding Environment

- C++ 20
- gcc-8 or recent
  - To check your gcc compiler version: $ g++ -v
- We will introduce three main ways of setting up the coding environment

  https://visualstudio.microsoft.com

C++ Coding Environment

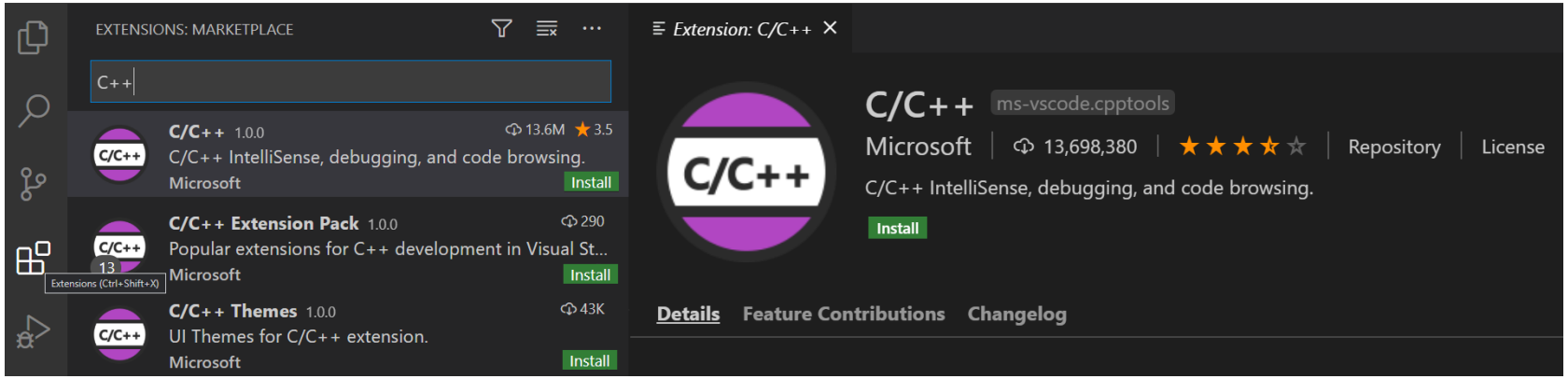
- C++ 20
- gcc-8 or r
  - To check
- We will int ent



C++ Environment Setup

method1: compile&run on cornell engineering linux server from terminal

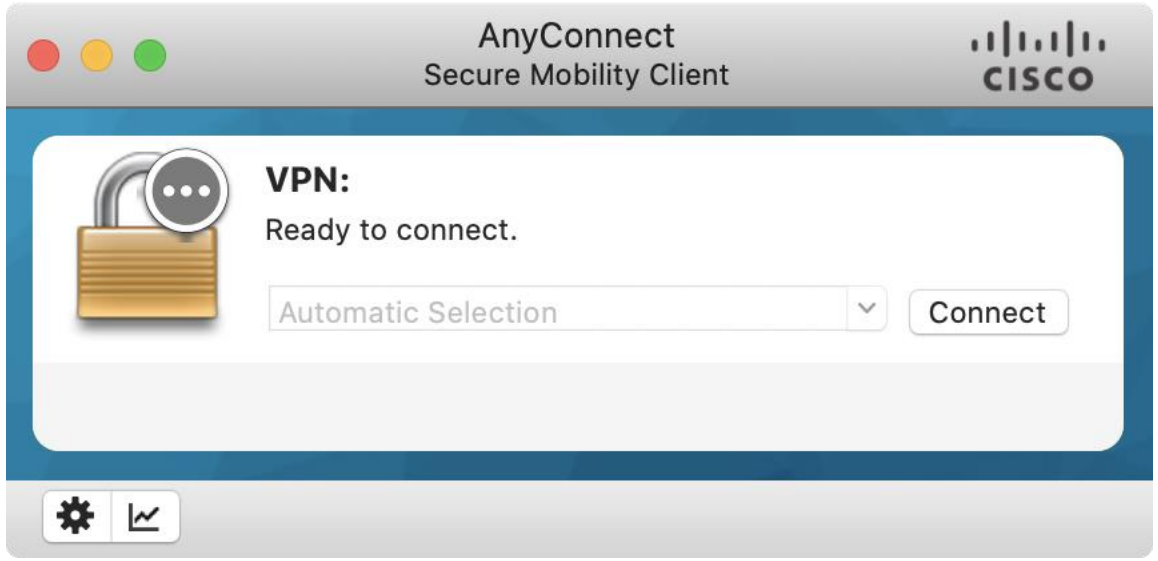
- Install Visual Studio Code
- Install C/C++ extension in VSCode



C++ Environment Setup

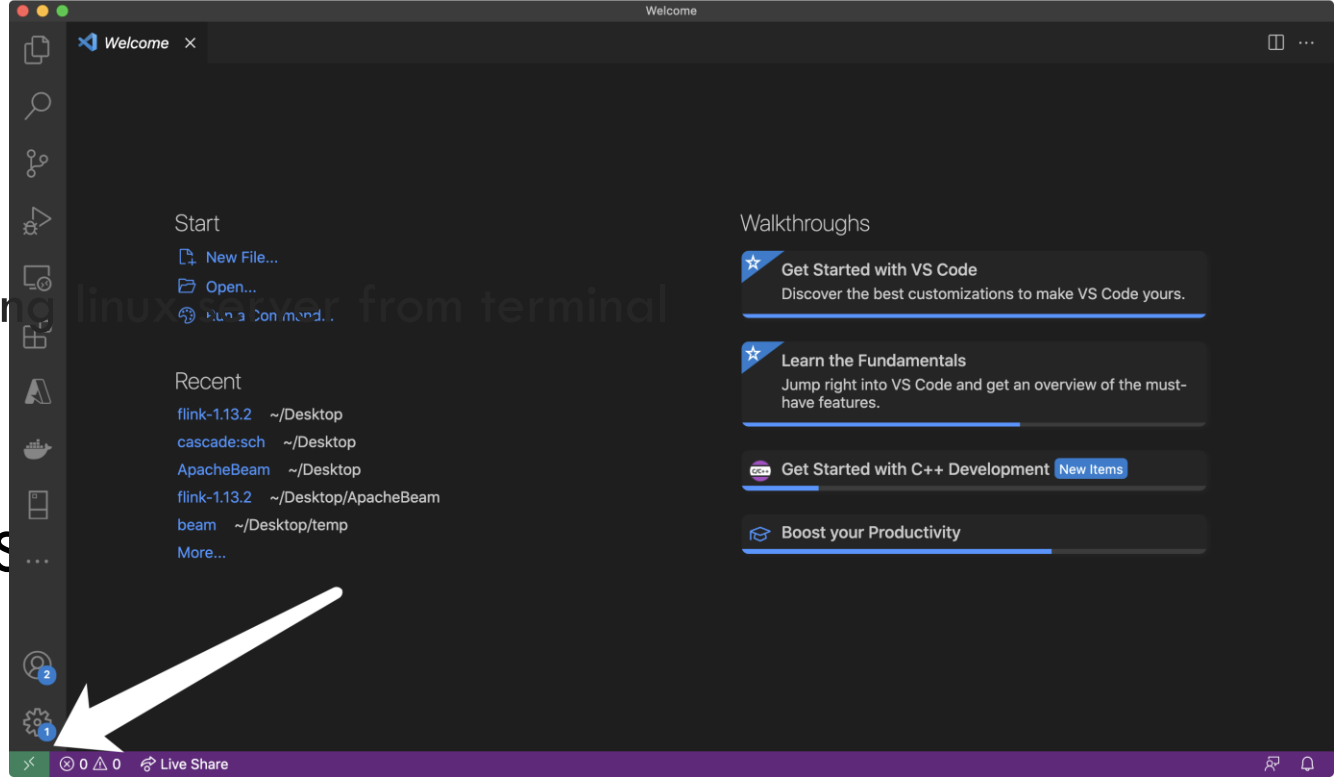
method1: compile&run on cornell engineering linux server from terminal

- Install VSCode and C++ extension on VSCode
- Login to Cornell VPN



C++ Environment Setup

method1: compile & run on cornell engineering linux server from terminal



- Install VSCode and C++ extension on VS
- Login to Cornell VPN
- ssh to your cornell student account on VSCode through
  - View -> Command Palette -> Remote SSH: Connect to host
  - In command type:

    % ssh [your netid]@ugclinux.cs.cornell.edu

C++ Environment Setup

demo

method1: compile & run on cornell engineering linux server from terminal

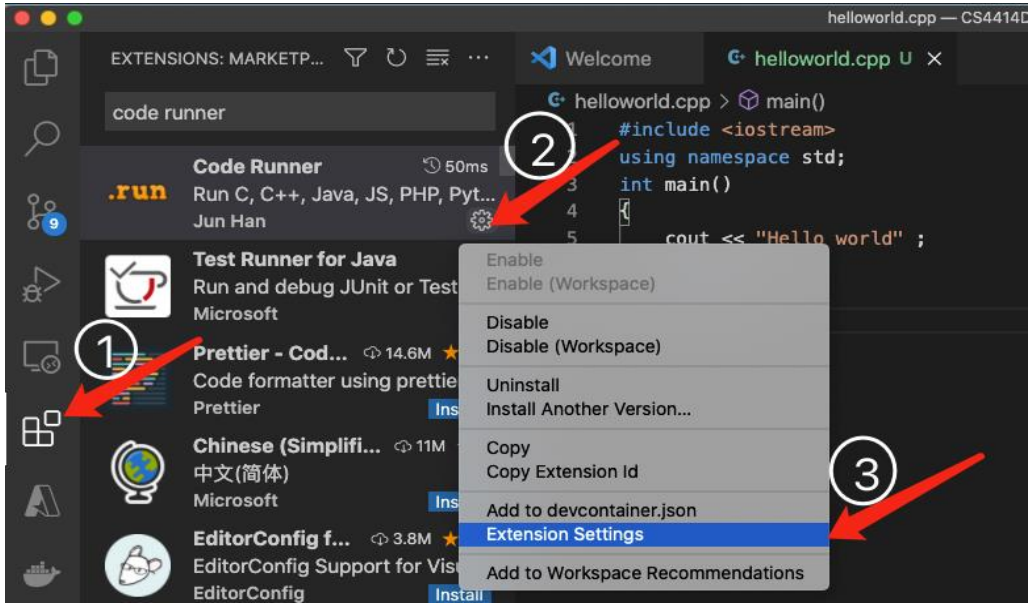
- Install VSCode and C++ extension on VSCode
- Login to Cornell VPN
- ssh to your cornell student account on VSCode through
- All set! Start coding (Demo)
  - Helloworld simple program
    - Compile: g++ -std=c++20 –Wall –o helloworld helloworld.cpp
    - Run: ./helloworld
  - **If there are multiple files, compile with:** g++ -std=c++20 main.cpp other.cpp etc.cpp

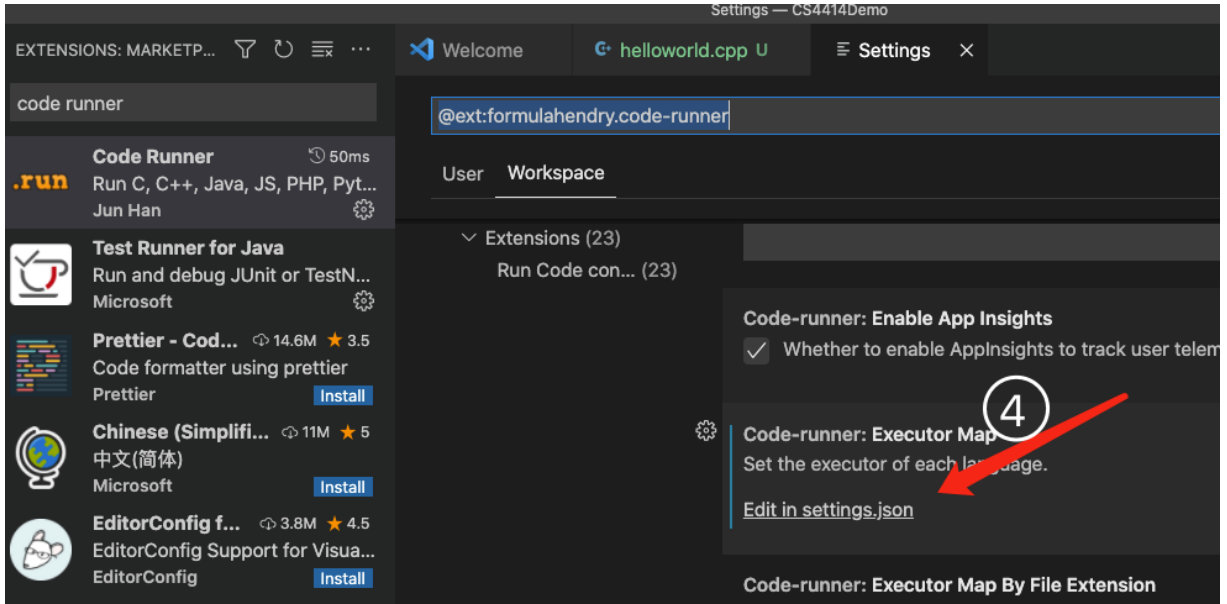
C++ Environment Setup

method2 : VSCode Edit, compile/run locally

1. Install IDE (VSCode in this example) , and extensions: C/C++, Code Runner

   - Specify C++ standard version on Code runner extension



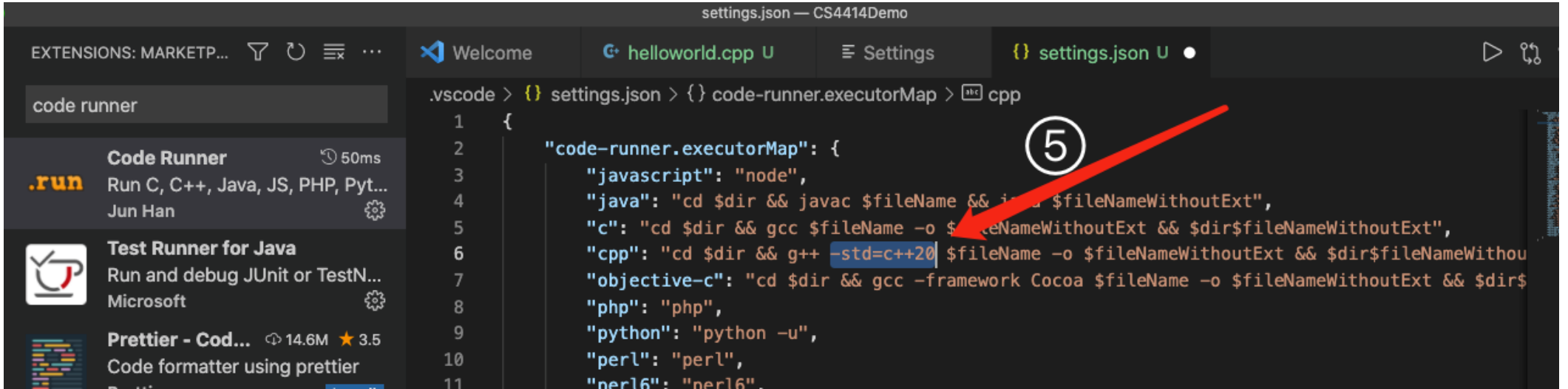


C++ Environment Setup

method2 : VSCode Edit, compile/run locally 

1. Install IDE (VSCode in this example) , and extensions: C/C++, Code Runner

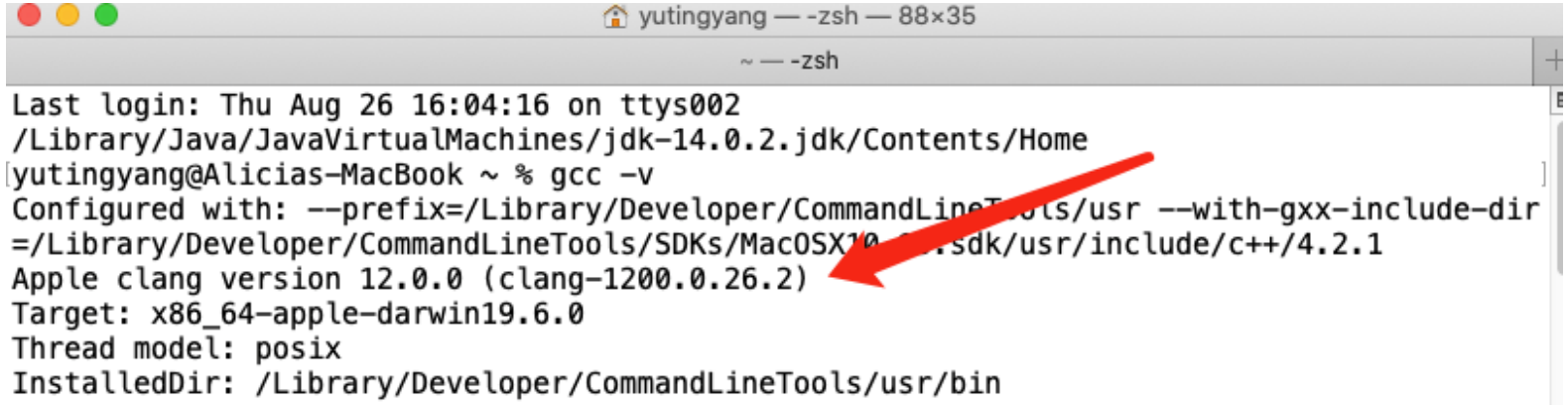
   - Add –std=c++17 to the json file:
     - "cpp": "cd $dir && g++ -std=c++17 $fileName -o $fileNameWithoutExt && dirfileNameWithoutExt"



C++ Environment Setup

method2: VSCode Edit, compile/run locally 

1. Install IDE (VSCode in this example) , and extensions: C/C++, Code Runner
2. Install Compiler
   - Why install gcc? mac default C++ compiler is Clang



C++ Environment Setup

method2 : VSCode Edit, compile/run locally 

1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner

2. Install GCC Compiler with following command

   % brew update

   % brew upgrade

   % brew info gcc

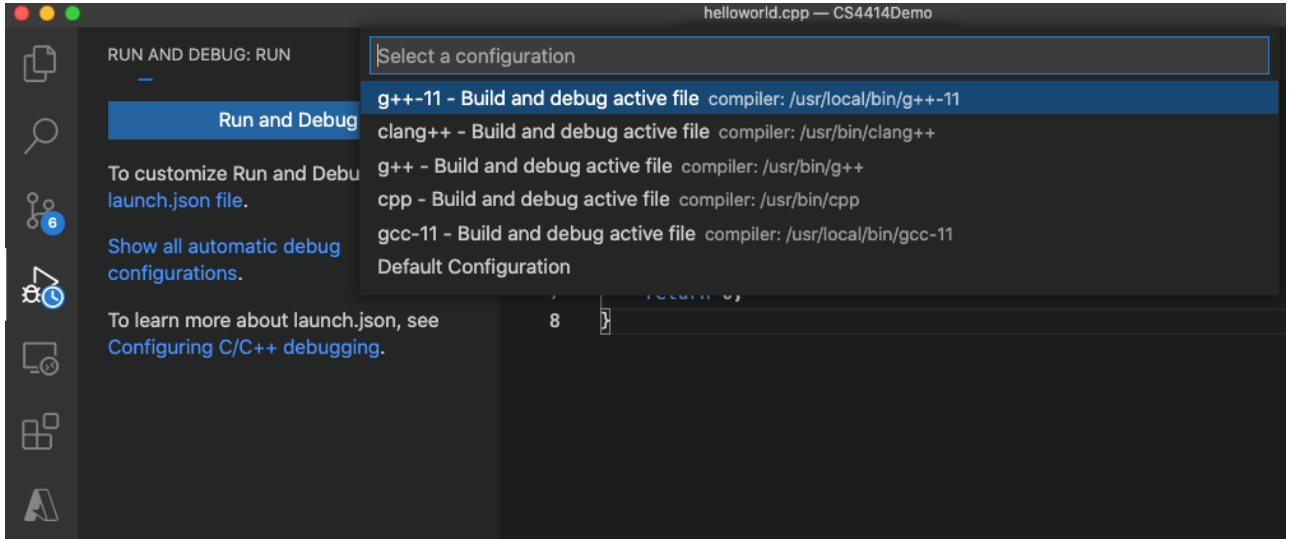
   % brew install gcc

   % brew cleanup

C++ Environment Setup

method2 : VSCode Edit, compile/run locally

- Install IDE (VSCode in this example), and extensions: C/C++, Code Runner
- Install GCC Compiler with following command
- Run and debug locally:
  - Control + shift + D



C++ Environment Setup

demo

method2 : VSCode Edit, compile/run locally 

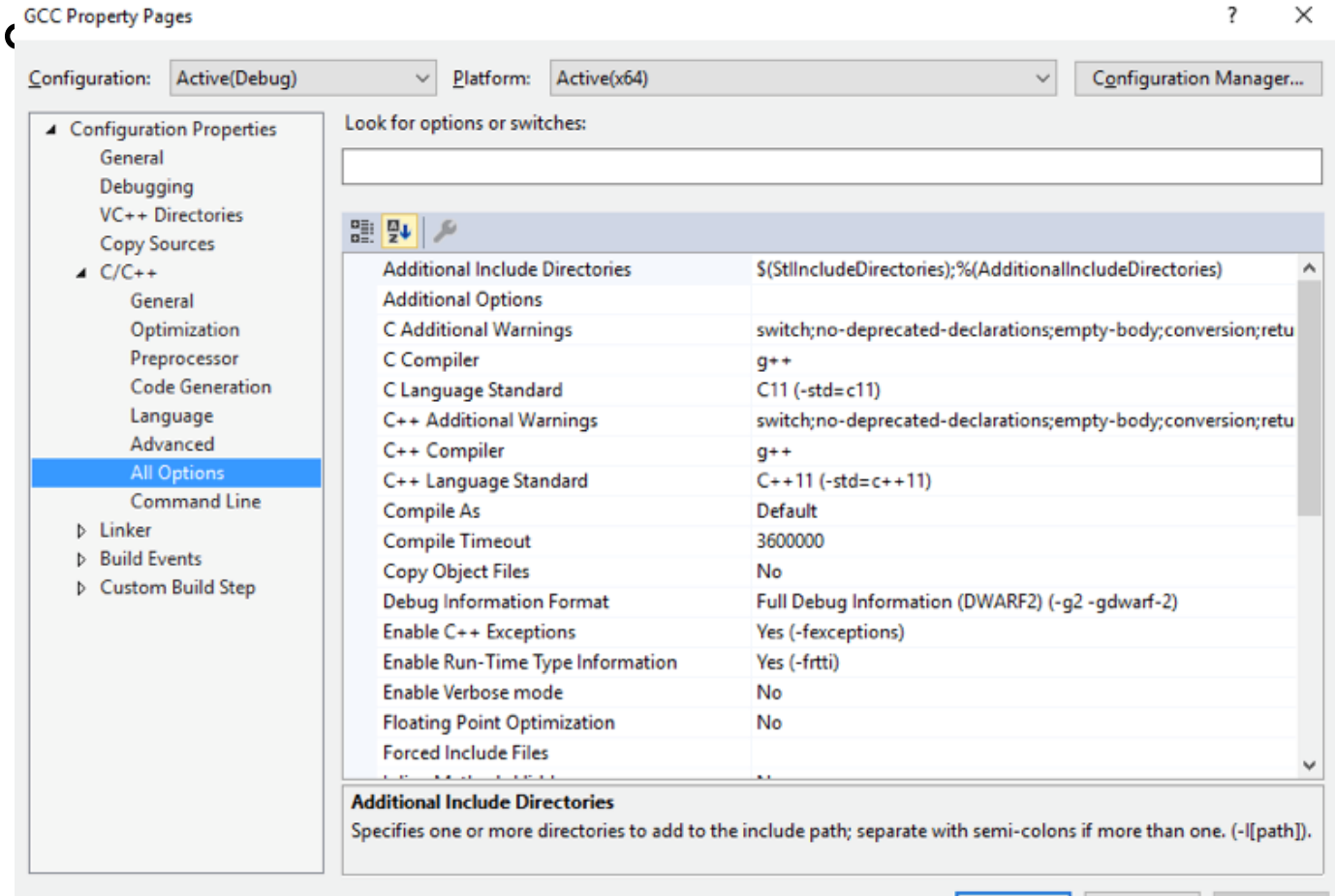
1. Install IDE (VSCode in this example), and extensions: C/C++, Code Runner
2. Install GCC Compiler with following command
3. Run and debug locally
4. Configure compiled file
   - .vscode/tasks.json

C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally

- Download Visual Studio

- Configure GCC property on Visual Stu



C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally 

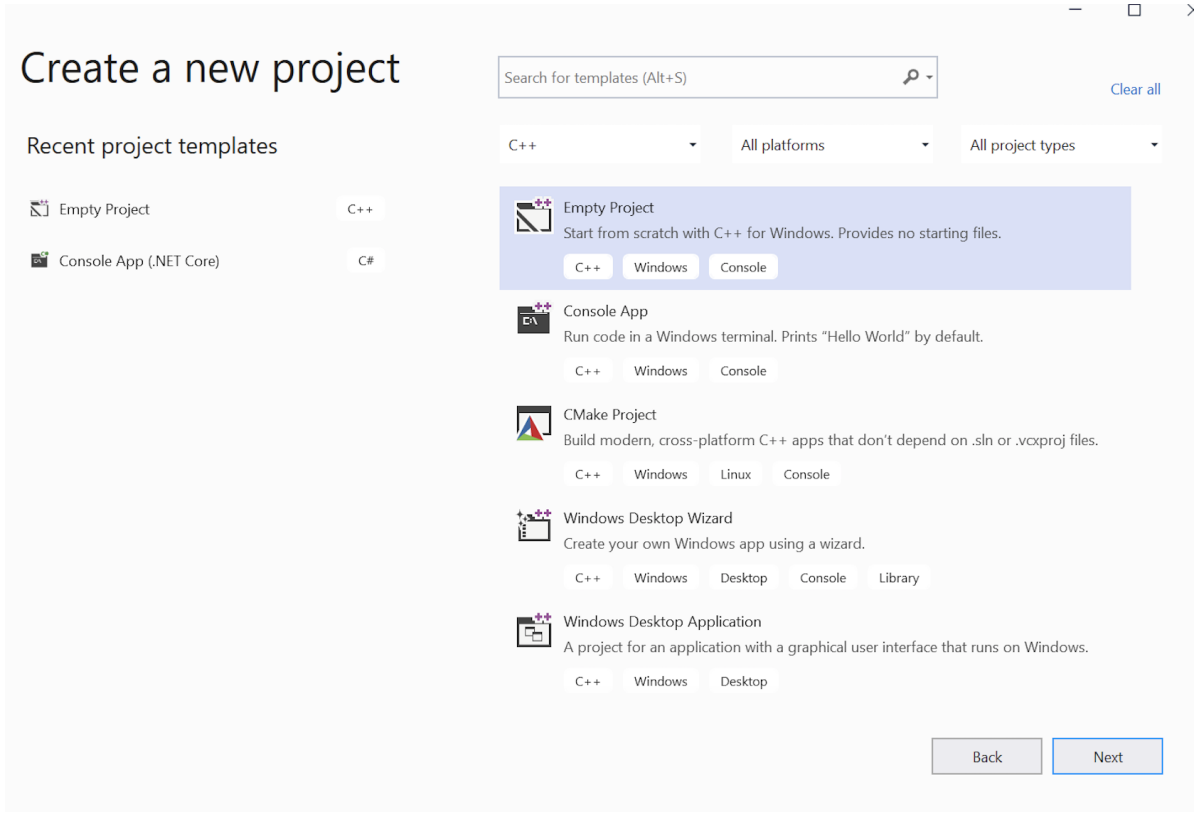
- Download Visual Studio
- Configure GCC property on Visual Studio
- Create C++ Project, right click project -> build

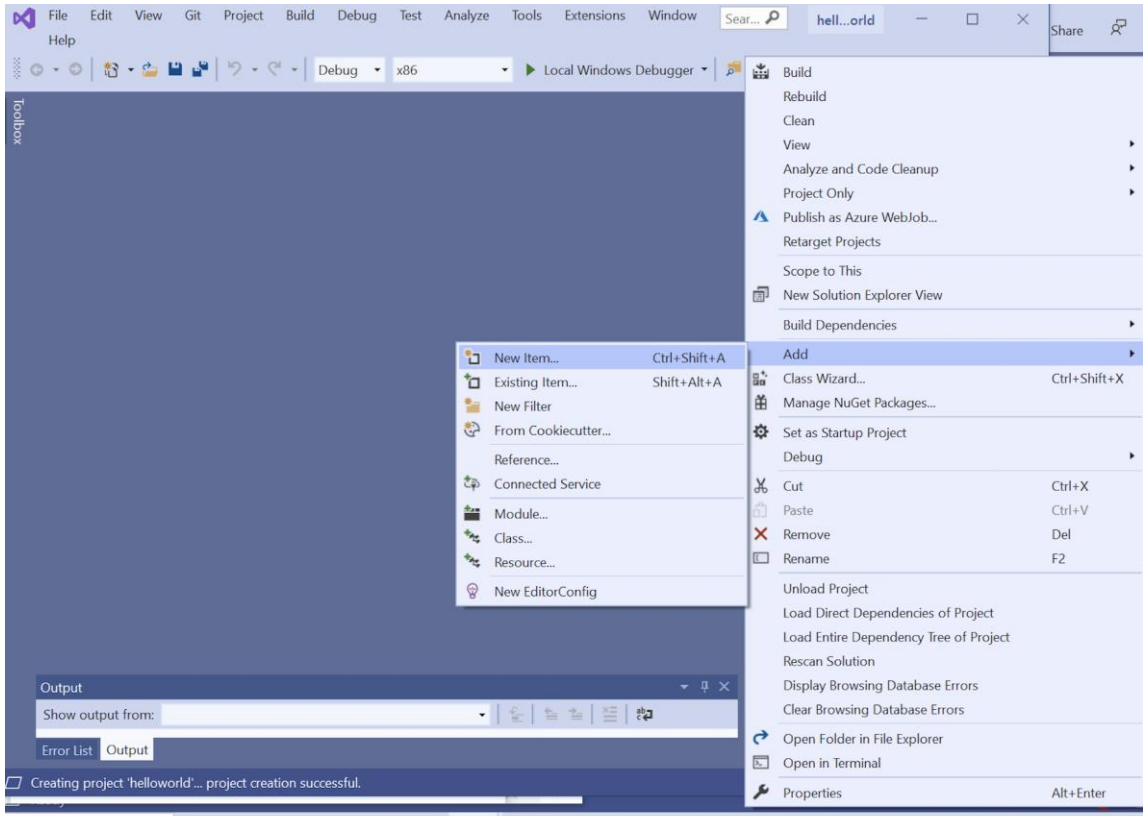
C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally

- Download Visual Studio

- Configure GCC property on Visual Studio

- 



C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally 

- Download Visual Studio
- Configure GCC property on Visual Studio
- Create C++ Project, right click project -> build
- Run the executable
  - Click the .exe
  - Click on localWindowDebugger on Visual Studio

C++ Environment Setup

- method3 (windows): Visual Studio Edit, compile/run locally