# Lab 1

# General Information

- Contact Information: Timothy Cui, jcui3@scu.edu
- Office Hours: TBD
- Rules
  - Attendance & demo are required.
  - Labs are due **before** the start of your next lab session. No late demos will be accepted.
    - 15% – 25% – 100%
  - No plagiarism
  - Do not post your code to Github
- Grading
  - 10% Attendance, 20% Style, 30% Demo, 40% Correctness

# Table of Contents

# C++ Compilation

- How to compile and run a C++ program
  - g++ <flags> <.cpp filename> -o <executable name>
  - ./<executable name>

# C++ Debugging

- How to debug a C++ program
  - Ex: g++ -Wall -Werror -g <cpp filename> -o <executable name>
  - gdb ./<executable name>
    - -Wall: enable warnings
    - -Werror: all warnings are treated as an error
  - Set breakpoint
    - break <line number>, break <function name>
    - continue[c], step[s], next[n]
      - Hitting “⬆” will lead to the previous command typed.
    - Conditional breakpoints
    - delete <breakpoint number>

# C++ Debugging

- How to debug a C++ program
  - Set watchpoint & print variables
    - watch <variable name>
    - print <variable name>
  - Quit gdb
    - quit[q]

# C++ Debugging

- Core File
  - If core file does not exist after a segmentation fault:
    - ulimit -a // Check core file size limit
    - ulimit -c unlimited // Remove the core file size limit
  - gdb ./<executable name> <core file name>
    - Ex. gdb ./a.out core.19761
  - To find more information about any commands, use man <command>
    - E.g. man ulimit
    - Press “q” to quit the man page

# C++ Debugging

- VSCode
  - Add breakpoints
  - Go over code line by line

# Program 1: Input Characters Counter

- Problem Description
  - **Task**: Given a string, output the number of alphanumeric characters and non-alphanumeric characters. Skip whitespaces.
  - **Input**: Your custom input in terminal
  - **Output**: “<input>” has ___ alphanumeric characters and ___ non-alphanumeric characters.
  - e.g.

# Program 2: String of Numbers & Reverse

- Problem Description
  - **Task**: The user enters a 10-digit number. The program should print out inputs and reversed inputs in a specified format (see below). If the input is illegal (size() != 10), notify the user and quit the program. You can assume that the input has no whitespace and only contains digits.
  - **Input**: Your custom 10-digit input in terminal
  - **Output**: 5 “tilted” rows of input and reversed input.
  - e.g.

```
[jcui3@linux20302 Lab1]$ ./p2
Enter 10 digits (no whitespace):
1234567890
     1234567890         0987654321
       1234567890         0987654321
         1234567890         0987654321
           1234567890         0987654321
             1234567890         0987654321
```

# Program 3: File Content Converter

- Problem Description
  - **Task**: Given a txt file, *after removing all non-alphabetical characters*, uppercase and print all words with length() >= 10. Your program should be able to **read the filename from terminal**. If the number of arguments from terminal is incorrect, print an error message and exit the program.
  - **Input**: A txt file
  - **Output**: Uppercased words with length() >= 10;

# Program 3: File Content Converter

- Problem Description
  - e.g

```
[jcui3@linux20302 Lab1]$ cat test.txt
1_counter.exe compiled successfully.
1_counter.exe got correct output.
Exit code: 0
2_pattern.exe compiled successfully.
2_pattern.exe got correcct output.
Exit code: 0
3_convert.exe compiled successfully.
3_convert.exe got correct output.
Exit code: 1
```

```
[jcui3@linux20302 Lab1]$ ./p3 test.txt
COUNTEREXE
SUCCESSFULLY
COUNTEREXE
PATTERNEXE
SUCCESSFULLY
PATTERNEXE
CONVERTEXE
SUCCESSFULLY
CONVERTEXE
```

# Other Resources

- ECC Remote Access
  - https://www.scu.edu/engineering/labs--research/labs/engineering-computing-center/connect-to-the-ecc-linux-platform-using-the-graphical-interface/
- Windows C/C++ Compiler Installation
  - https://www.freecodecamp.org/news/how-to-install-c-and-cpp-compiler-on-windows/
- C++ Tutorial
  - https://www.w3schools.com/cpp/

# Deliverables

- Three .cpp files, with each completing one program
  - Do not submit other files
  - Each .cpp file should start with a header (name, program description, school ID, contact info)
    - See below
  - Make sure your code can run on school linux machine

```
/**
 * Name: Timothy Cui
 * Program Description: The program is the server side of the project. It
 *   will get client's form, make some analysis, and return the prediction
 *   result back to the client
 * School ID: 1574417
 * Contact Info (Optional): Add me on Discord
 */
```