Debugging & Unit Tests

Complete this worksheet by answering the questions using the given program. Make sure to color or **bold** your answers so they are distinguishable from the question.

Getting Started

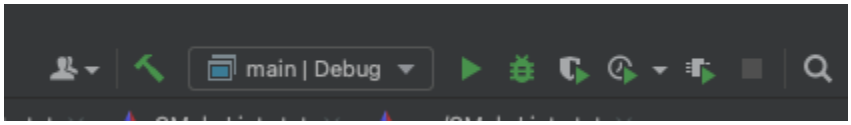
Use the starter code linked in the Canvas assignment description. Open the project using CLion.

Part 1

Complete this section by looking at the code that has the comment `//Part 1`, included in the following files:

- src/debugger.cpp
- src/debugger.h
- src/main.cpp
- tests/debugger_test.cpp)

Run the program with main as the executable file. Do this by choosing **`main`** in the dropdown menu next to the play button. It should look like this:



1. You should see errors in the console output/messages screen. Look in the main.cpp file. Which function is being called? MultiplyTheLongWay

   There are a couple of problems with the code. Even if you immediately see the issues, walk through these steps and answer these questions to practice some problem-solving skills together.

   We already know the function's name, now let's get a little more familiar with it by looking at the prototype in the debugger.h file.

2. What is the return type? int

3. What are the parameters? int leftOperand, int rightOperand

4. What is the namespace it belongs to?namespace debugger

5. Are we missing anything from the prototype? If you can't tell do the following steps and check each item off as you do: it is missing a semicolon at the end;

    - ☑ ~~Look at the output screen and read the top error. If you still can't find the error, move to b.~~

    - ☑ ~~Ask yourself if the prototype has everything it needs to be complete and correct.~~
        - Return type
        - Name
        - A place for parameters ()
        - ; at the end

6. What kind of error was this: syntax, logic, or data? syntax

7. Add the correction to the code and run the program again. What is printed? 5

   The program should successfully compile now. However, there is one more error in the code. Answer the following questions about the function in order to find it.

8. Look at the function definition in debugger.cpp. What is the code supposed to do?It multiplies 2 integers 'the long way' which is through addition. It's using a for loop to add the left operand to itself as many times as the right operand specifies.
9. Look in main.cpp and observe the call to `MultiplyTheLongWay()`. Based on the input, is the output to the console correct? What should it be?No, it should be 35.
10. Place a debugging point in debugger.cpp next to the function definition header for `MultiplyTheLongWay()`and run the debugger. What variables are currently in memory and what are their values? leftOperand = 5 and rightOperand = 7

11. Walk through the function slowly by pressing the 'step into' arrow until the cursor ends up at the end of the function. How many times does the code in the loop body get executed?One time
12. How many iterations should we see our loop complete? (*Hint*: Look at the for loop condition) the loop condition is i < rightOperand so we should see 7 loops

13. Take a look at how the for loop is set up. Does it have everything it needs to be correct and complete? Check off the box after you have looked and confirmed.

    - ☑ ~~Is the `for` keyword used?~~

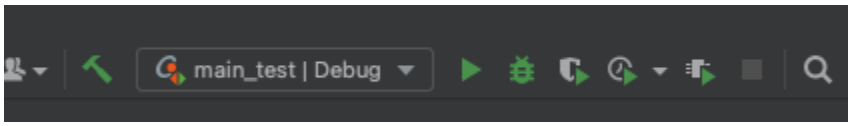
    - ☑ ~~Does the loop have both the opening and closing parenthesis i.e., for(...)~~

    - ☑ ~~Is the control variable correctly initialized *(hint: int i = 0;)*?~~

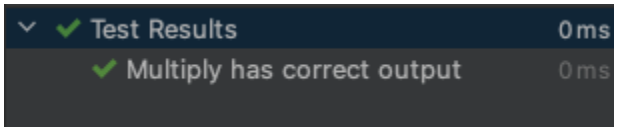
    - ☑ ~~Is the condition set up correctly *(hint: i < rightOperand;)*?~~

- [x] ~~Is the control variable incremented? *(hint: i++;)*~~
- [x] ~~Are there opening and closing curly braces, do you need them here?~~
- [x] ~~Does the code in the curly braces look correct for continuously adding the leftOperand into a result variable?~~

14. What is the error? What kind of error is this: syntax, logic, or data?
    There is a semi-colon at the end of the for loop declaration. This is a syntax error
15. Add the correction to the code and run the program again. What is printed?
    35
    Now that you have verified the code works in main, let's test it using unit tests. Switch the executable from main to main_test. It will look like this:



16. Looking at the debugger_test.cpp file, look at the test listed under the `//Part 1` comment. There are two CHECK statements. Read the code in between parenthesis of the CHECK tests. What is each CHECK testing checking for? They are checking that 5 * 7 outputs 35 and 3 * 9 outputs 27.
17. Run the unit tests by pressing play. Your output should look like this:



Part 2

Complete this section by looking at the code that has the comment `//Part 2` above it. Before you start, uncomment everything that has the `//Part 2` label over it in the following files:

- src/debugger.cpp
- src/debugger.h
- src/main.cpp
- tests/debugger_test.cpp)

TIP: To comment or uncomment large blocks of code, first highlight everything you want to modify. Then press **`ctrl + /`** for Windows or **`cmd + /`** for MacOS.

Make sure to switch the executable file back to **`main`** before starting, as well.

1. Comment out `//part 1` in main.cpp. Run the program. Which function is being called and what is the output for the total?The function being called is CalculateOrderTotal()

2. Look at the function prototype for `CalculateOrderTotal()` in the header file. Does it accept any parameters? What are they? Yes, an int called userChoice;
3. Look at `CalculateOrderTotal()` being called in main.cpp again. Are any arguments passed into the function? Why is option 4 still chosen from the menu?No, no arguments are passed into the function, but thre reason 4 is chosen is because in the function prototype userChoice was assigned 4, just in case an int isn't passed in
4. Observe the function definition for `CalculateOrderTotal()` in the debugger.cpp file and examine the values for the variables named `hamburger`, `fries`, and `drink`. Option 4 should add all of these variables together and apply a 20% combo discount. Is the console output for the total correct? No.

5. Now look at the switch statement. Which case handles the combo meal output? Case 4

   Debugging Time: Put a breakpoint under the '`case 4:`' statement (should be line 47) and run the debugger. Once compiled and the cursor is at the breakpoint, click the "step into" button once. This should step you into the `CalculateComboDiscount()` function. If it didn't, the debugger may have taken you to a page you don't recognize. If so, press "step out". This should bring you back to debugger.cpp. Now, choose "step into" again. Do this until you are taken to the `CalculateComboDiscount()` function.

6. Look at the variable section in the debugger output. What does the variable called `amount` have stored in it? It has a double 5.97
7. Click "step into" again. Look at the variable section in the debugger output. What does the `discount` variable have stored in it? Is this correct? Discount has 1.19. Yes, it is correct. 20% of roughly 6 is 1.20.
8. Click "step into" again. Look at the variable section in the debugger output. What does the `newPrice` variable have stored in it? Is the discount applied correctly? 7.16, no. it is adding the discount amount instead of taking it away.
9. Is the discount being applied correctly? Correct the code to apply the discount and run the program again. What is your output this time? No, it's being added to the total instead of taken away. 4.78 is the output after correcting the code.

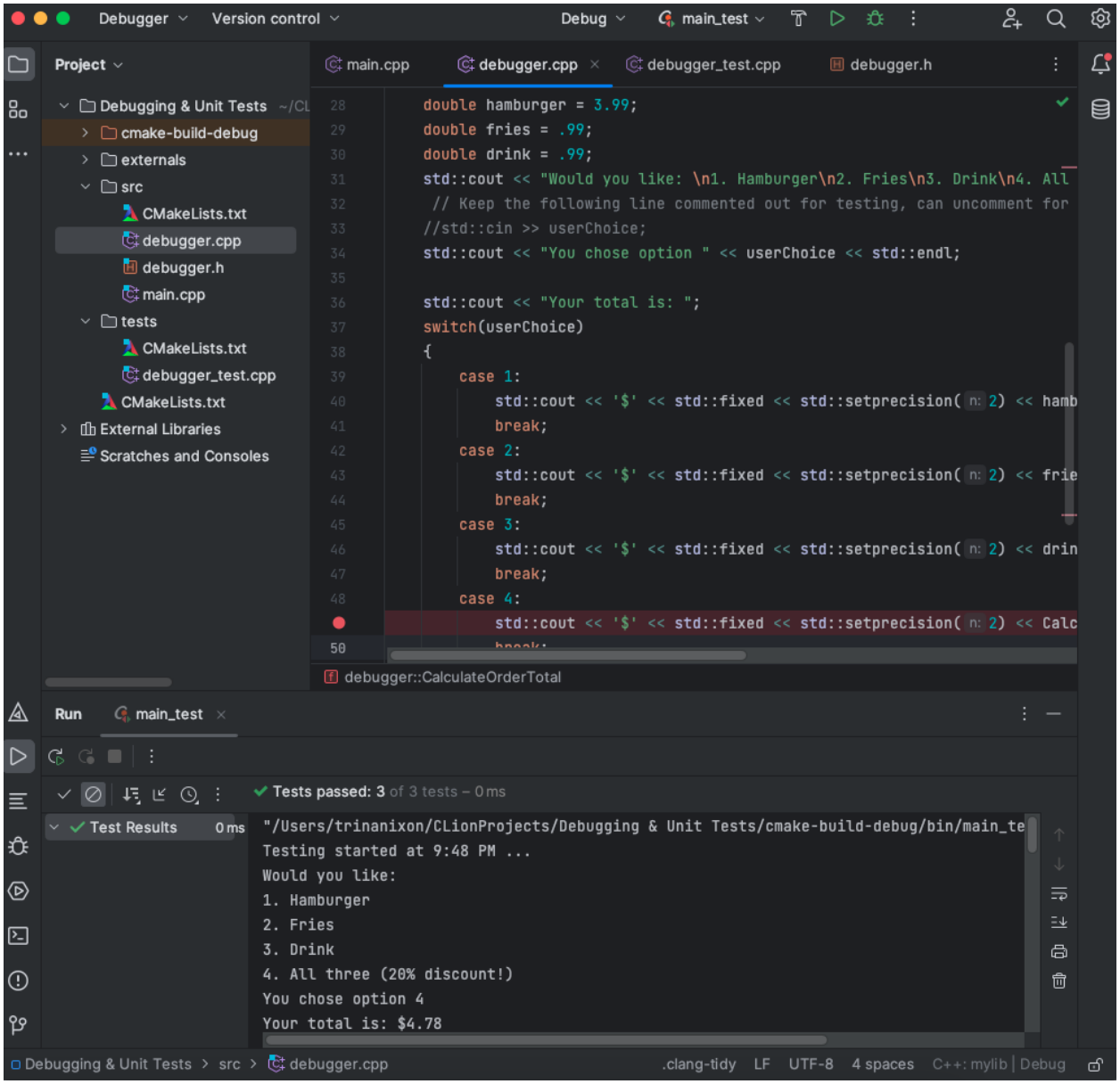
   Now that you have verified the code works in main, let's test it using unit tests. Switch the executable from main to main_test. Make sure both sets of tests are uncommented under the comment //Part 2

10. One way to set up unit tests is to ask functions to return a "true" value if everything executed correctly and a "false" value if something went wrong.

The `CalculateOrderTotal()` function returns true as long as a valid option (1-4) is chosen. Run the unit tests. Examine the output and determine which unit tests are failing. What kind of argument is being passed into CalculateOrderTotal() and is causing the unit tests to fail? The test without an argument and the test with the argument 4 are both failing. So passing in no argument or the value 4 is causing the function to return 'false' instead of true.

11. Go to the switch statement and observe the code for case 4. Since 4 is a valid option, check to see if anything is missing from the code that would cause the function to return false rather than true.(*hint*: remember how switch statements work if there is no break statement after each case is finished) it is missing the break statement so it goes to the default case which returns 'false.
12. Add the missing statement to the code. Run the tests again. Paste a screenshot here of the passing tests.

.


Debugger
Version control
Debug
main_test
Project
Debugging & Unit Tests ~/CL
cmake-build-debug
externals
src
CMakeLists.txt
debugger.cpp
debugger.h
main.cpp
tests
CMakeLists.txt
debugger_test.cpp
CMakeLists.txt
External Libraries
Scratches and Consoles
main.cpp
debugger.cpp
debugger_test.cpp
debugger.h
double hamburger = 3.99;
double fries = .99;
double drink = .99;
std::cout << "Would you like: \n1. Hamburger\n2. Fries\n3. Drink\n4. All
// Keep the following line commented out for testing, can uncomment for
//std::cin >> userChoice;
std::cout << "You chose option " << userChoice << std::endl;
std::cout << "Your total is: ";
switch(userChoice)
{
case 1:
std::cout << '$' << std::fixed << std::setprecision(n: 2) << hamb
break;
case 2:
std::cout << '$' << std::fixed << std::setprecision(n: 2) << frie
break;
case 3:
std::cout << '$' << std::fixed << std::setprecision(n: 2) << drin
break;
case 4:
std::cout << '$' << std::fixed << std::setprecision(n: 2) << Calc
50
debugger::CalculateOrderTotal
Run
main_test
Tests passed: 3 of 3 tests – 0 ms
Test Results 0 ms
"/Users/trinanixon/CLionProjects/Debugging & Unit Tests/cmake-build-debug/bin/main_te
Testing started at 9:48 PM ...
Would you like:
1. Hamburger
2. Fries
3. Drink
4. All three (20% discount!)
You chose option 4
Your total is: $4.78
Debugging & Unit Tests > src > debugger.cpp
.clang-tidy LF UTF-8 4 spaces C++: mylib | Debug