Instructions

For this programming assignment, you will implement a program in assembly that will determine if integer values entered by the user are prime. To do this, you will convert the provided C++ code (below) to MIPS Assembly Language using the QtSPIM simulator. There are two different versions of the code you can implement. The first version allows for the user to enter a single value, the primality is determined, then the program terminates. The second allows for the user to enter as many values as they would like before terminating. The second version is slightly more complex, so converting that code instead of the standard version will award you 5 bonus points. A starter .asm file is provided for you. All code for this project is to be written in a single .asm file titled "CMPS_351_Assembly_PA1_ULID.asm" where ULID is replaced with your ULID. Once completed, upload the .asm file to Moodle for your submission.

Grading

The final assembly code is worth 70 points. Relevant and proper documentation throughout the file is worth 30 points. This program should be easy, however you must be sure to set it up properly or you will face difficulties. Accordingly, there should be significantly more documentation and comments in your code than you would traditionally write for a C++/Java project. I suggest placing a comment roughly every other line (or even to the right of every line) stating what you are intending the line/code to do (like the samples in the slides are). This will greatly simplify the debugging process for you should you encounter problems and aid the grader in evaluating your assignment. The more you help the grader understand what you are trying to do, the easier it will be for them to grade your assignment making them happier and your grade higher (most likely).

As always, documentation should be present as follows. Any and all variables that are not obvious by their name, should include a comment (located on the line above or same line) detailing what that variable is for. Same goes for any code segments that are not directly obvious in their intent. Any group of code doing a common thing should have a comment. If you are performing multiple steps for something that is not clear and obvious, put a comment above the line where each step begins saying what will happen. As a general rule: *If you are unsure whether a comment should be there or not, put one.*

Header

Include the following header at the top of your .asm file. You may not substitute the certificate of authenticity for your own version. Code submitted without a certificate of authenticity will not be graded. If the certificate is missing, the grader will email you individually (with the instructor CC'd) to reply with a certificate for the assignment. If you do not reply to this email with a certificate within a week of the request being sent, you will receive an automatic zero on the assignment and the grade cannot and will not be changed. You will only be contacted with the request once.

```
/*
 * Name: {Your Name}
 * ULID: {Your ULID}
 * Course: CMPS 351-{Your Section}
 * Assignment: {Assignment Name}
 * Certificate of Authenticity: (choose one)
 * I certify that the code in the method function main of this project
 * is entirely my own work.
 * (or)
 * I certify that the code in method function main of this project is
 * entirely my own work, but I received assistance from [insert name].
 * Follow this with a description of the type of assistance.
 */
```

Code to Convert - First Version

```
#include <iostream>

using namespace std;

int main() {
    int value = -1;
    cout << "Welcome to the Prime Determiner!" << endl
        << "Enter a number and I will tell you if they are prime." << endl;

    cout << "Please enter a positive integer: " << endl;

    bool isPrime = true;

    for (int i = 2; i < value; i++) {
        if (value % i == 0) {
            isPrime = false;
            break;
        }
    }

    cout << (isPrime ? "That value is a prime!" : "That value is not a prime!") << endl;

    cout << "Thanks for using my program!" << endl;

    return 0;
}
```

Bonus Points Code - Second version

You can earn 5 bonus points by converting this code instead of the code above that simply allows for multiple numbers to be tested before the program terminates.

```
#include <iostream>

using namespace std;

int main() {
    int value = -1;
    cout << "Welcome to the Prime Determiner!" << endl
        << "Enter some numbers and I will tell you if they are prime." << endl;

    while(true) {
        cout << "Please enter a positive integer (-1 to quit): " << endl;

        cin >> value;
        if(value == -1)
            break;

        bool isPrime = true;

        for (int i = 2; i < value; i++) {
            if (value % i == 0) {
                isPrime = false;
                break;
            }
        }

        cout << (isPrime ? "That value is a prime!" : "That value is not a prime!") << endl;
    }

    cout << "Thanks for using my program!" << endl;

    return 0;
}
```

Documentation/Reflection (30 pts)
Remember that the standard relevant documentation throughout the assignment is worth 30 points of the project's grade.

As part of your documentation, exercise a growth mentality by reflecting on this assignment and your work. Feel free to say whatever you want, but you are required to answer the following in a block comment at the very **bottom of your .asm file**. You are graded completion of this, not what you say.

- How many hours did you spend on this assignment?
- What was the most difficult part for you? Why?
- What was the easiest? Why?