Homework 2: Prime Number Project

AMS 562: Introduction to Scientific Programming in C++

Due: **Mon, Sept. 29, 2025 at 11:59 PM**

Purpose

This homework is designed to reinforce your understanding of conditional statements, loops, and functions in C++. You will also practice working with multiple source files, header files, and Makefiles. The project is focused on prime number computations, a classical problem in computer science and mathematics.

This assignment involves writing a program that performs several prime number-related tasks, such as checking if a number is prime or finding the prime factors of a given number.

Tasks

The homework consists of four main tasks. Each task will be implemented in C++ using the provided skeleton code. Your implementation should follow the structure of the skeleton code and complete the TODOs marked in the code.

- **Task 1: Divisibility Check**
  - Write a function `is_divisor` that takes two integers and checks if the second integer is a divisor of the first.
  - **Hint:** Use the modulus operator (`%`) to determine if a number is divisible by another.
- **Task 2: Prime Number Check**
  - Write a function `is_prime` that takes an integer and returns whether it is prime.
  - **Hint:** To determine if a number is prime, check divisibility from 2 up to the square root of the number.
- **Task 3: Prime Factorization**
  - Write a function `print_prime_factors` that prints the prime factors of a given integer.
  - **Hint:** Use a loop to divide the number by its smallest divisor repeatedly until it reduces to 1.
- **Task 4: Print First N Primes**
  - Write a function `print_primes` that prints the first n prime numbers.
  - **Hint:** Use the `is_prime` function in a loop to find prime numbers and print them.

Skeleton Code

Here is a brief explanation of the skeleton code. You will find `TODO` comments where you need to complete the code.

ams562_hw2.cpp

This file contains the `main()` function that will call the different tasks based on user input. The TODOs in this file are mostly related to reading user input and calling the correct functions. The program will loop over tasks until you input `0` to exit.

prime.h

This header file declares all the functions that need to be implemented in `prime.cpp`. Complete the TODOs by adding function prototypes.

prime.cpp

This file contains the function definitions. Complete the TODOs by implementing the functions. Be sure to follow the algorithm hints provided in the tasks section.

Makefile

The provided Makefile will compile `ams562_hw2.cpp` and `prime.cpp`. You will need to ensure it compiles without errors and links all the object files correctly.

Batch Testing Instructions

We have provided a set of test input and reference output files on Brightspace in a zip file under the `test` folder. For batch testing, follow these steps:

- Compile your project as usual using the `make` command.
- Run the program with input redirection and output redirection:

```
./ams562_hw2 < test_input.txt > test_output.txt
```

- Compare the generated `test_output.txt` with the provided `reference_test_output.txt`.

During grading, we will use a different set of test inputs to evaluate your code. Ensure that your code handles edge cases and follows the specification provided.

Test Cases

Below are some sample test cases to help you verify your code. Ensure that your program produces the expected output for each case.

Input	**Expected Output**	**Task**
12, 4	4 is a divisor of 12	Task 1
12, 5	5 is not a divisor of 12	Task 1
17	17 is prime	Task 2
18	18 is not prime	Task 2
18	Prime factors: 2 3 3	Task 3
5	2 3 5 7 11	Task 4

Table 1: Sample test cases for the homework assignment.

Submission Instructions

You are required to use GitHub for this assignment. Follow these steps:

- Download the skeleton code from Brightspace and unzip it to your local machine.
- Create a private repository on GitHub.
- Complete the TODOs in the provided skeleton code.
- Add a `README.md` file in your repository that briefly describes your code, how to compile it, and how to run it.
- Push your changes to your private GitHub repository.
- Add the TA (GitHub username `yuxuanye1`) as a collaborator to your private repository so that they can access it for grading.
- Submit the link to your private GitHub repository through Brightspace.

Files to Submit

Ensure that your repository contains the following files:

- `ams562_hw2.cpp` – Your main program file.
- `prime.cpp` – Your function implementations.
- `prime.h` – Your function declarations (header file).
- `Makefile` – A Makefile to compile the project.
- `README.md` – A brief description of the code, compilation, and usage instructions.

Grading Rubric

Your submission will be graded based on the following criteria:

- **Correctness (50%)**: The code correctly implements the functionality for each task and passes all test cases.
- **Code Structure (20%)**: Your code is properly organized into header files, source files, and a Makefile.
- **GitHub Usage (20%)**: You use GitHub effectively, with clear commit messages and appropriate version control.
- **Code Style (10%)**: Your code is well-documented with comments, and it follows proper C++ conventions.

Hints for Algorithm Design

- For prime number checking, you only need to check divisibility up to $\sqrt{n}$ for a number n.
- For prime factorization, you can use trial division: start with 2 and continue dividing by the smallest divisor until the number becomes 1.
- In the task to print the first n prime numbers, maintain a count of the primes found and use the `is_prime` function to verify each number starting from 2.