## I. Introduction

There will be no final examination in this course. The course project is therefore an important requirement; and it counts 30% towards your final grade.

The project is to be done by teams. Each team will be formed by at least two (2) but no more than three (3) students. A list of students in the class is available on Canvas. Please form your team and send your team formation to me so I can set up a group for your team. An optional discussion board has been set up to help students to form a team. Please note that it is your responsibility to form a team. If you are not able to form a team, you can still do your project on your own, but you will lose a significant number of points. The same also applies if you have formed a team but then all team members except you have withdrawn from the course, leaving you as the only remaining team member.

## II. About the Project

Select one language from the following list:

- C (but not C++)
- C#
- Clojure
- Erlang
- F#
- Fortran
- GO
- Haskell
- Io
- JavaScript (with Node.js)
- Julia
- Kotlin
- O'Caml
- Pascal
- Perl
- R
- Ruby
- Scala
- SmallTalk
- Visual Basic

You will then do a thorough investigation of the selected language, with emphasis on its unique features compared to other languages. You will also need to implement a program (details described below) using the (same) language you have selected. You need to tell me which language you select by Sunday October 13. If you prefer, you can also suggest your own project idea. Student suggested projects need to be approved by the instructor.

## III. Programming

You need to implement a program to demonstrate the Goldbach conjecture. The German mathematician Christian Goldbach (1690-1764) conjectured that every even number greater than two can be represented by a sum of two prime numbers. This conjecture has never been proved or disproved – although it has been verified for values up to $4×10^{18}$ in 2012. As such, you may assume it is true for the cases considered in this project. The goal of the program you are going to implement is to find ***all*** unique ways to represent a given even number as a sum of two prime numbers. A prime number is an integer greater than one that is evenly divisible by only itself and 1. The first few prime numbers are 2, 3, 5, 7, 11, …. There may be several ways to represent a given even number as a sum of primes. For example, the even number 26 may be represented as 3 + 23, 7 + 19, or 13 + 13. The output of the program should contain a summary line, telling how many Goldbach pairs can be found for a number. It is then followed by a list of lines, in the form of $n = p + q$, where $n$ is the input to the program, and $p$ and $q$ are the two prime values, with $p \le q$. The lines should be sorted in ascending order of $p$ (or equivalently, descending order of $q$). For example:

```
We found 3 Goldbach pair(s) for 26.
26 = 3 + 23
26 = 7 + 19
26 = 13 + 13
```

In order for you to be able to demonstrate your program, your program should also have a main function (or equivalence) that allows you to read data from a file (so you can run your Goldbach function for multiple values).

For example, the Python example provided (see later) can be invoked using one of the following commands:

```
python goldbach.py
python goldbach.py data.txt
```

For the purpose of demonstration, you can assume the input parameter to the Goldbach function is never greater than 100,000.

A Python program has been provided to demonstrate how your program should function. Please note that you may need to make appropriate adjustments to fit the programming framework of your selected language.

> Challenge: There is an inefficiency in the Python sample program provided. This inefficiency makes it run relatively slow (over 90 seconds to finish processing the given input data file on a 6th generation i5 core laptop). Can you find out and fix the inefficiency? (A more efficient Python implementation should increase the performance by at least 100% and take less than 40 seconds to process the given input data file on the same machine.)

Your program should demonstrate the followings:

General Requirements:

1. Functional decomposition (into functions, subroutines, methods, etc.) – that is, your program should ***NOT*** be just a big slump of code.
2. Documentation – you should add comments at key positions to explain what you want to do.
3. Readability – you should utilize indentation (and/or other mechanisms) to ensure/enhance the readability of your code.

Operational Requirements:

4. File handling – your program should be able to read data from a file.
5. Output – your program should be able to produce output to the screen.
6. Array/list handling – your program should be able to store values to / read values from an array or a list.
7. Command line argument handling – your program should be able to handle argument(s) passed in on the command line.

In particular, the following are the features that ***must be*** found in your program in order to fulfill the operational requirements:

- Your program should be able to compute the Goldbach pairs for one or more input values (R4).
- For each input value, your program should first output a statement reporting how many Goldbach pairs have been found, ***before*** outputting the found Goldbach pairs. That means you will need to first store all found Goldbach pairs so you can count them before outputting (R5 and R6).
- You should output one line for each pair of the found Goldbach pairs, in the form of <input value> = <p> + <q>, with p <= q. The lines should be arranged in increasing order of <p> (R5).
- When giving no command line argument, your program should output the Goldbach pairs for the input values of 3, 4, 14, 26, and 100 as a simple demonstration (R7).
- When given one or more command line arguments, your program should assume that the first argument to be the name of an input file, and process all numbers stored in that file. Additional arguments can be ignored (R7).
- You can assume that the input file always exists, and it contains data in the correct format (that is, no error checking is needed).
- You can assume that the input file contains one number per line, and those numbers are always even integers between 4 to 100,000 (inclusive). The number of lines in the file is NOT known in advance, and the end of the file signals the end of input. Also, the values in the file can appear in any order (R4).

## IV. Mid-Project Consultation

You can make an appointment with me during Week 12 (Nov 04-10) to have me look at your project and give comments. This consultation is optional and can be done either in-person in my office or virtually through MS Teams. If you choose to do it, and if you are working as a team, both team members must be present during the consultation meeting. Please book a time slot for the meeting through the following Microsoft Bookings link[1]: https://outlook.office365.com/owa/calendar/AlbertChan@uncfsu.edu/bookings/s/YYZs1-sxoEeqxaK8_HIRdw2. (You can also use the QR code at right to access the booking page.) Bookings must be made in advance with a lead time of no less than 24 hours and no more than 14 days. Only one member of a team needs to make the booking. However, please communicate the booking information to your team partner. Please note that this link is NOT the same links you will see later for project presentation.

## V. Communication Evidence

Since this is a team project, it is important that you demonstrate communication among team members. The communication must be documented in the group discussion board on Canvas – and you must demonstrate ***on going communication*** – that is, you cannot just pour everything in during the last week. I need to see continuous communication happening every week after your team is formed, and ***all team members*** are participating in communication.

## VI. Final Report

You should prepare a final report with grammatically correct English about their project. By Friday November 29, each team must submit an 8- to 12-page (not including the title page and the bibliography pages) final report. The final report must be written in grammatically correct English. It should include (at least, but not limited to) the following sections:

- Introduction to the project

[1] Note that the booking pages say the booking is for CSC451. However, the same link is also applicable to this course.

- Detailed information
    - A brief history of the language
    - Notable properties of the selected language
    - Information about implementation the Goldbach conjecture program
    - Other information that you see fit
- Conclusion
- Bibliography – Please note that you must have at least five (5) items in your bibliography list and at least sixty percent (60%) of your bibliography must be in printed form (that means they should not be ***ONLY*** available on the Internet). Examples of such items include textbooks, technical reference books, journal articles, conference proceedings.
- DO NOT include program code in the final report – if you think you should include the code for completeness, you should put it in an appendix and those pages do not count toward the 8- to 12- pages.
- Minor usage of AI-assisted writing tools (such as Chat GPT or Google Bard) is acceptable. But the AI-generated contents should not be more than 10% of your report, and their usage should be clearly marked and/or annotated.
    - The usage of AI-generated contents needs to be reported.

Format for Final Report

- Fonts: 12-point (Except for the Headings and title page)
- Spacing: double spacing
- Margin: 1-inch on each side
- Paper: standard letter-size paper
- Fancy title page and/or figures are NOT necessary.

## VII. Presentation and Demonstration

At the end of the semester, each group must perform a 20-minute presentation and demonstration about their project. Each member of the team must participate in the presentation and demonstration (that means each member must be presenting – they cannot just be responsible for changing the slides or other similar supportive tasks). The presentation should involve using presentation tools such as PowerPoint or similar software. The demonstration should be at the end of the presentation and should last about 5 minutes.

Presentations can be performed virtually through Microsoft Teams® on Tuesday Dec 3 or Wednesday Dec 4; or in-person in my office on Thursday Dec 5. The links[1] for booking in-person presentation and virtual presentation are: https://outlook.office365.com/owa/calendar/AlbertChan@uncfsu.edu/bookings/s/a8PjPkCjP0u2LrtsPztZWg2 and https://outlook.office365.com/owa/calendar/AlbertChan@uncfsu.edu/bookings/s/86zBz8qmsUOfvhH0NpDeVw2, respectively. (You can also use the QR codes below to access the booking page.) Bookings must be made in advance with a lead time of no less than 24 hours and no more than 14 days. Please note that these links are NOT the same link you saw on the earlier for mid-project consultation.

QR Code for In-person Presentation

QR Code for Virtual Presentation

Hints for the presentation:

- Contents of the presentation are more important than fancy effects.
- Your presentation must feature the parallel programming aspects of your project.
- Do not cramp too much information into a single slide (e.g., you should not use fonts smaller than 18 points).
- Control the number of slides to match the presentation time. (A quick guideline is to reserve one minute for each slide. However, you should always rehearse your presentation in advance to ensure your presentation is neither too long nor too short.)
- Make sure the slides stay on the topics you want to deliver to your audiences.
- Assume your audiences are just lay people. (That is, you should not assume that your audiences already have a deep knowledge about parallel programming – however, you can assume that they have a good understanding about programming – e.g., a sophomore student majoring in Computer Science.)
- Cue cards during the presentation are acceptable but you should not depend on them.

## VIII. Evaluation

The project will be evaluated as follows:

- Final report – 12 points – graded based on structures, readability, and the contents of the report – with 2 points on Spelling and Grammar.
- Program code – 4 points – graded based on program structures and readability (general requirements R1 to R3). The baseline is that I should be able to understand how your code works and see it basically works correctly ***without*** actually executing the code.
- Presentation – 5 points – graded based on presentation skill, see above.
- Demonstration – 4 points – graded based on the successfulness of the demonstration, which in turn is based on the features and correctness of your code (operational requirements R4 to R7).
- Communication Evidence – 5 points – graded based on how strong the evidence about the teamwork that has been carried on within the team throughout the timespan of the project.

## IX. Submission

| Element | Due Date | Submission Format | Submission Location | Late Penalty |
|---|---|---|---|---|
| Group Formation | Sunday October 13 | Email | Email | All project points forfeited |
| Final Report | Friday November 29 | Word Document or PDF | Canvas | 50% deduction on the Final Report grade |
| Program Code | | Fully commented code in the selected programming language | | 50% deduction on the Program Code and Demonstration grades |
| Presentation PowerPoint | | PowerPoint or PDF | | 50% deduction on the Presentation grade |
| Communication Evidence | On Going | Discussion Board Messages | Discussion Board via Canvas | |

In any case, all components must be submitted by Sunday December 01.