Math 248: Computers and Numerical Algorithms, Spring 2025
Project One: The Goldbach Conjecture
Handed out 2/19, Due 3/26

Project Teams: You may work in teams of **up to three** people. Working in groups is not required, but if you do at most three in a group, and each group should submit one project. I'll make sure everyone involved gets appropriate credit.

Specifications: Your project will consist of three parts:

1. **A written description of the work.** This will constitute approximately half your grade, and can be written in any text editor you like. Word is fine, but if you know LaTeX, use it. The following outline should be followed for each part.

   (a) **A title page.** This should at minimum include a title, date, and the names of the group.

   (b) **A statement of the problem in your own words.**

   (c) **A description of the solution procedure.** You should state the methods used to solve the problem, identifying any special aspects of the implementation. An english outline of any algorithms used should be included here, including any relevant mathematical discussion. If any equations are required, include them on separate lines with equations numbers, which makes referencing them in the text particularly easy. Pictures, tables or graphs may be included as appropriate.

   (d) **Discussion of results and conclusions.** Include relevant results in an appropriate form, which may include tables, graphs or other relevant pictures. Explain your results and discuss what their relevance may be. Address any troubles or difficulties you encountered.

2. **Computer Programs.** The other half of your grade will be for the Matlab functions you produce to form your solutions. The text should be included in your submission, as well as submitted separately. The names of all members of your team should be included in each function, as well as the standard required elements in the homework assignments so far. Functions will be graded first on performance (do they work), and second on programming style.

3. **Honor Pledge.** The final part will be a statement of the pledge that the material is your group's work. Any additional sources used should be cited. All members of each group are expected to contribute to the project, and I expect you to include a brief summary of the contributions of various group members to the project.

Final thoughts: Projects are expected to be challenging, and should take you a large amount of time to complete. To minimize all-nighters close to the deadline, it is essential to get started as soon as possible. Note that as well as developing programming skills, technical writing skills are an important aspect of these projects. Please proofread your work, and read each other's contributions to make sure that the point is clearly made.

1. A positive integer greater than one is called *prime* if its only divisors are one and itself. One is not considered prime. The first few are $2, 3, 5, 7, 11, 13, \ldots$, and there are an infinite number of them. Write a Matlab routine called `prime` that takes as input a positive integer and outputs one if it is prime, zero if it isn't. A B-grade function will have a loop that tests all the integers from 1 to the integer, and an A-grade function will use as few applications of `mod` as possible. Hint: think about how factors come in pairs.

2. Using your function `prime`, write code (not a function) to find out how many primes there are between two and 100 000. How high can you increase the upper bound in a reasonable time?

3. Deciding whether a number is prime by looking for potential factors is relatively slow. A more efficient technique is known as the Sieve of Eratosthenes. Start by writing out all the numbers from one to n, and cross out one, which is not prime. The process at every step is to circle the first unlabeled number, then cross out all multiples of that number. Keep going until the first unlabeled number is greater than or equal to the square root of n. At this point every circled or unlabeled number is prime. For example, (you should do this on paper), write out from one to fifty ($\sqrt{50} \approx 7.07$). Cross out 1. Then circle 2 and cross out all its multiples, $4, 6, 8, \ldots, 50$. Then circle 3 and cross out all its multiples, $6, 9, 12, \ldots, 48$. Then circle 5 and cross out $10, 15, 20, \ldots, 50$. Then circle 7 and cross out $14, 21, 28, \ldots, 49$. The next unlabeled number is 11 which is greater than $\sqrt{50}$, so we are done, and the prime numbers can be read off as $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47$.

   Write a Matlab function called `listprimes` that takes as input a positive integer n and outputs a one dimensional array of length n where the ith element is one if i is prime, zero if it is not. This is equivalent to starting with an array of all ones, and zeroing elements instead of crossing them out. Use this function to verify how many primes there are between two and 100 000, counting ones in the array. Comment on the speed compared to your code in question 3. How far can you push the upper bound this time?

4. In 1742, Christian Goldbach conjectured in a letter to Leonhard Euler that every even number ≥ 4 can be written as the sum of two prime numbers. This has never been proven, despite lots of attention, and is one of the oldest and most famous unsolved problems in mathematics. Brute force checking has verified the Goldbach conjecture up to 4×10^{18}.

   Write a Matlab function called `goldbach` that takes as input the array produced by `listprimes` as well as an even number n, and outputs the number of distinct pairs of primes that add up to it. For example, $26 = 3 + 23 = 7 + 19 = 13 + 13$, so the output in this case would be 3. Make sure you input an array holding prime number data that is sufficiently big – for a given n we should generate an array from `listprimes` that is at least of length n.

   Note that you could go through every pair of primes and check whether their sum is n, but that is inefficient. I recommend a single loop through all the primes $\leq n/2$, and check whether the difference between n and each prime is itself prime. For example, with $n = 26$, the primes are $2, 3, 5, 7, 11, 13$ and the differences are $24, 23, 21, 19, 15, 13$, and three of these are in our list as prime numbers.

5. Using your function `goldbach`, write code (not a function) that produces an array holding how many pairs of distinct prime factors every even number from 4 to 10 000 has. Make a plot of the data, and see if you can see any patterns in it. Hint: only store results for even numbers in the array, so you don't have lots of zeros screwing up your plot. Write code to find out how many numbers have exactly one pair of distinct prime factors. How many have exactly two or three? What is the largest number of pairs, and for what number?

6. *Open Ended Bonus Question* (only if you found the above very easy): Twin primes are pairs of primes that differ by two. The first few are $3, 5, 7, 11, 13, 17, 19, 29, 31$.

   (a) Write a function called `listtwin` that outputs a one dimensional array where the ith element is one if i is a twin prime, zero if it is not. I recommend starting with the code in `listprimes`, and come up with a way of zeroing out the non-twin primes.

   (b) Repeat the Goldbach conjecture question using only twin primes. You should be able to find that only 33 even numbers below one million can't be represented by a sum of a pair

of twin primes, but most can. Virtually no work has been exploring this area that I know of.