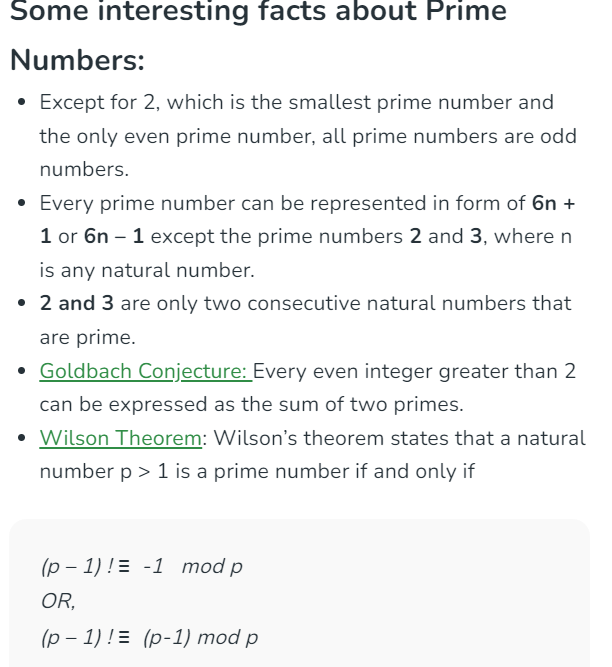
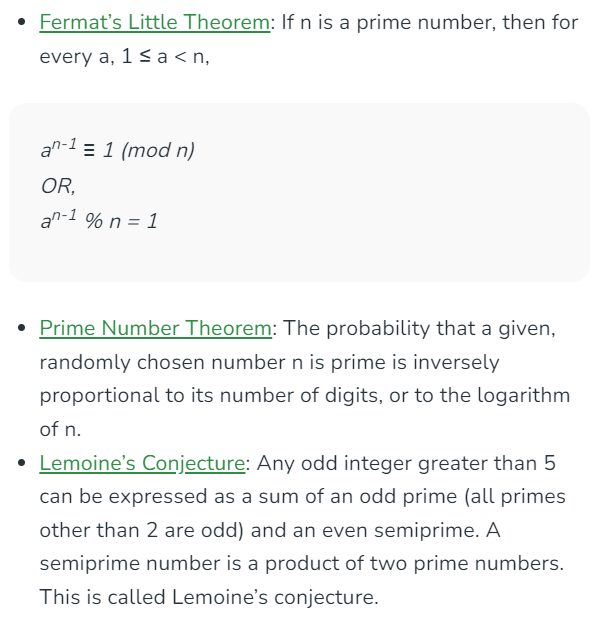
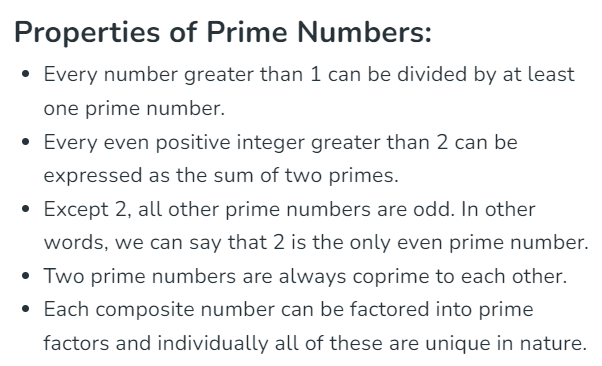
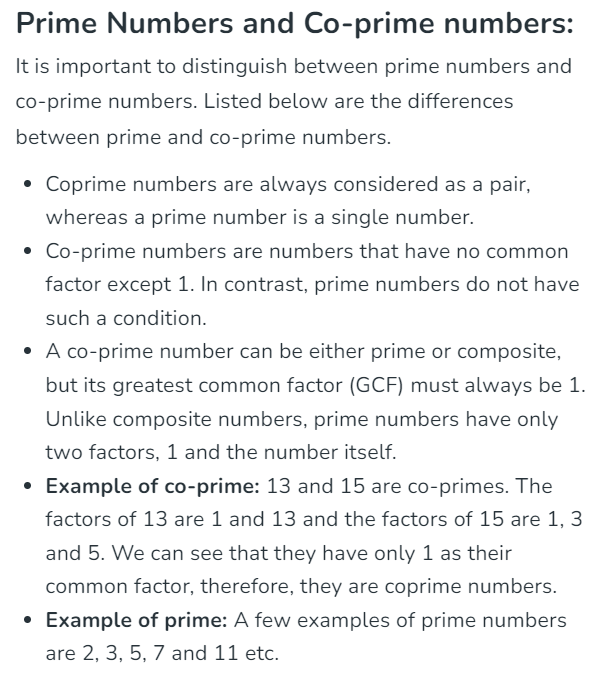
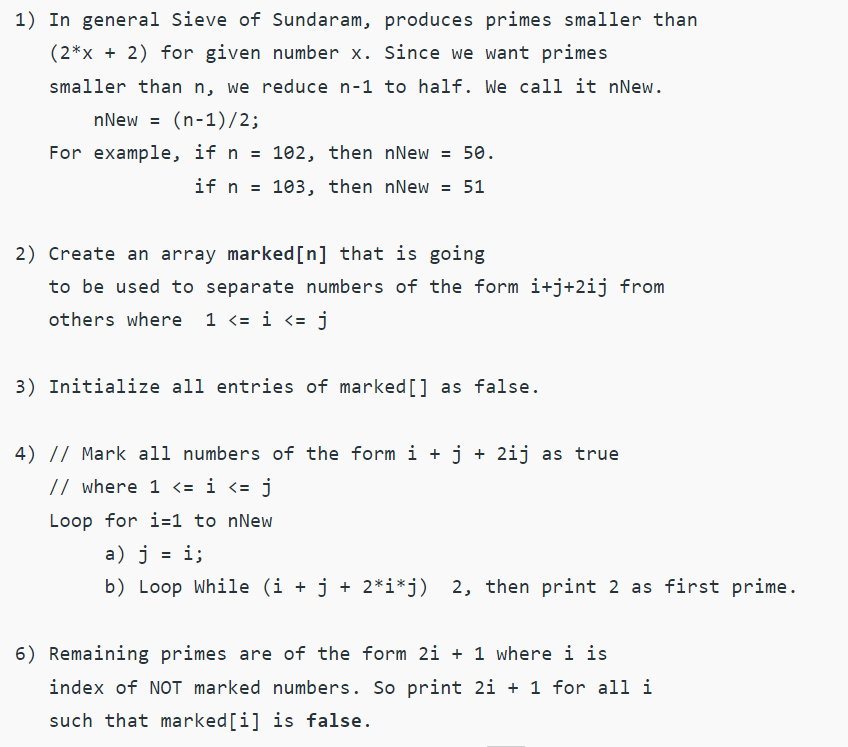
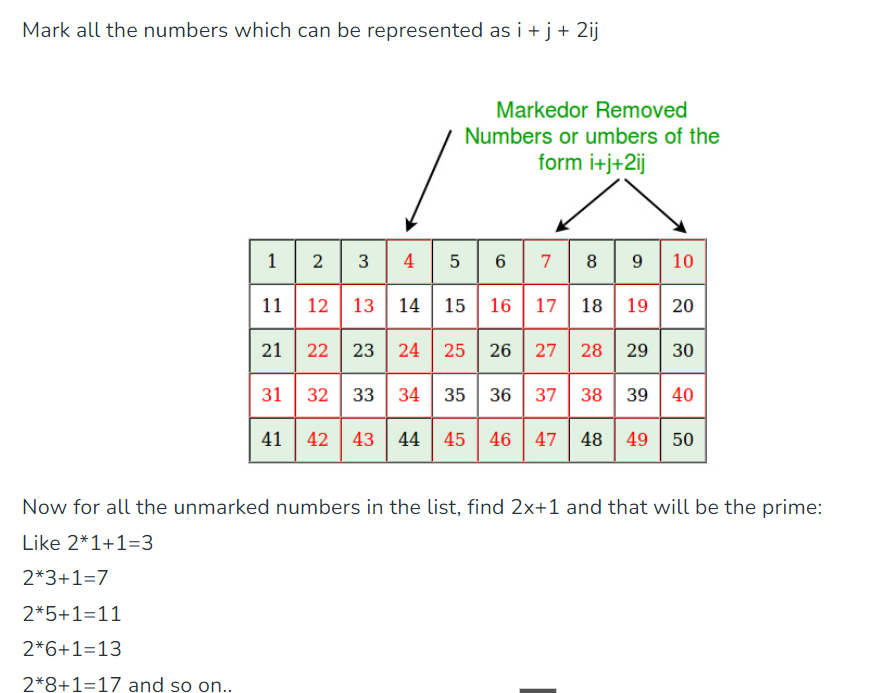
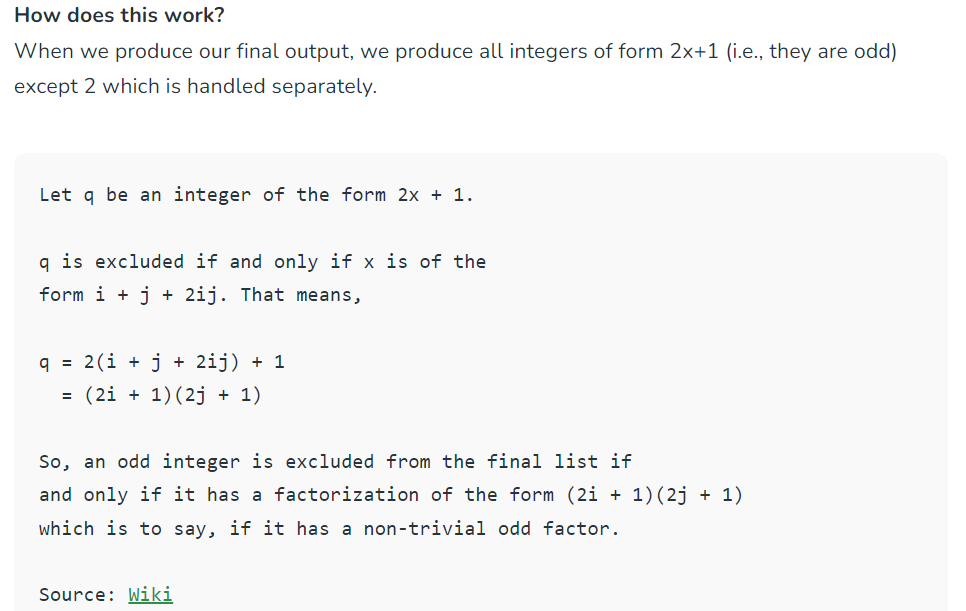
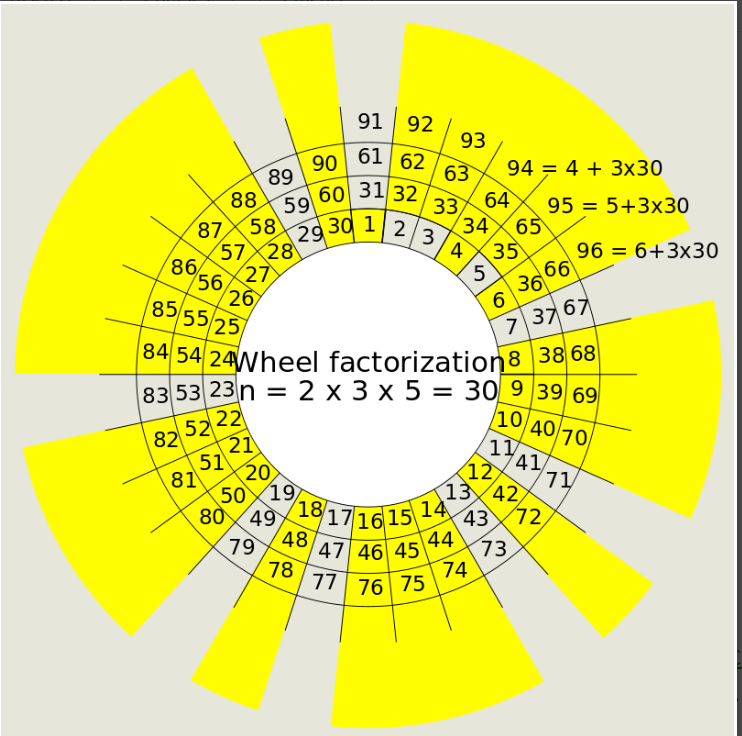
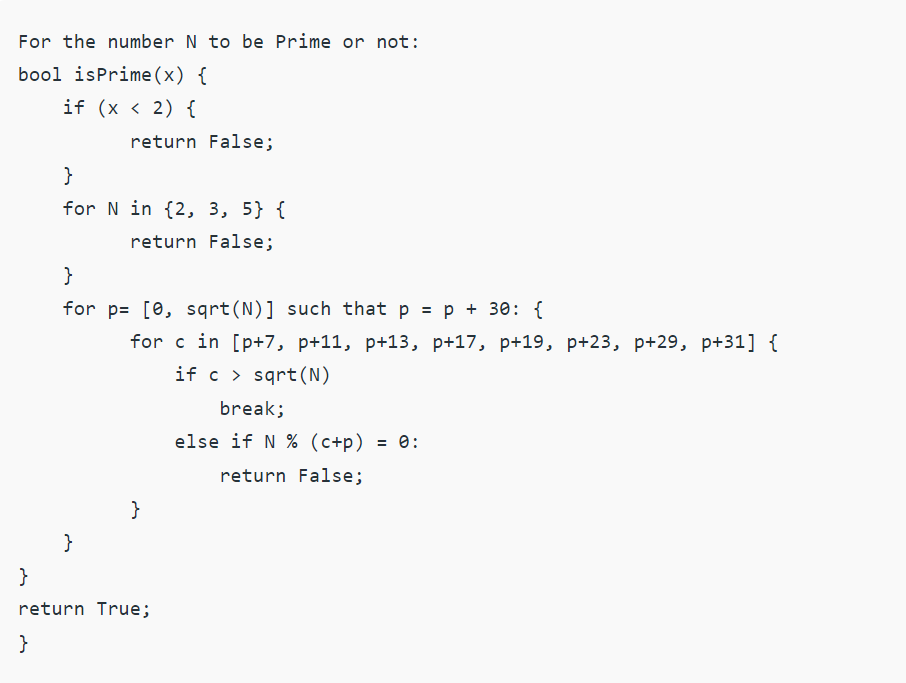
Prime Number & Primality Tests

1. 
2. 
3. 
4. 
5. Sieve Of Eratosthenes
   1. The sieve of Eratosthenes is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million or so (Ref [Wiki](http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes)).
6. Segmented Sieve
   1. The idea of a segmented sieve is to divide the range [0..n-1] in different segments and compute primes in all segments one by one. This algorithm first uses Simple Sieve to find primes smaller than or equal to ?(n). Below are steps used in Segmented Sieve.
      1. Use Simple Sieve to find all primes up to the square root of ‘n’ and store these primes in an array “prime[]”. Store the found primes in an array ‘prime[]’.
      2. We need all primes in the range [0..n-1]. We divide this range into different segments such that the size of every segment is at-most ?n
      3. Do following for every segment [low..high]
         1. Create an array mark[high-low+1]. Here we need only O(x) space where **x** is a number of elements in a given range.
         2. Iterate through all primes found in step 1. For every prime, mark its multiples in the given range [low..high].
   2. In Simple Sieve, we needed O(n) space which may not be feasible for large n. Here we need O(?n) space and we process smaller ranges at a time (locality of reference)
7. Sieve Of Sundaram
   1. 
   2. Time Complexity:O(nlogn)
   3. Auxiliary Space: O(n)
   4. 
   5. 
8. Bitwise Sieve – To Do
9. Sieve of Atkin – To Do
10. Wheel Factorization Algorithm
    1. Given a number **N**. The task is to check if the given number is [Prime Number](https://www.geeksforgeeks.org/prime-numbers/) or not.
    2. Wheel Factorization is the improvement of the method [Sieve of Eratosthenes](https://www.geeksforgeeks.org/sieve-of-eratosthenes/). For wheel factorization, one starts from a small list of numbers, called the **basis** — generally the first few [prime numbers](https://www.geeksforgeeks.org/prime-numbers/), then one generates the list, called the **wheel**, of the integers that are coprime with all numbers of the basis. Then to find the smallest divisor of the number to be factorized, one divides it successively by the numbers on the basis, and in the **wheel**.
    3. 
    4. Let say we select **basis** as **{2, 3, 5}** and the numbers which are coprime to the basis are **{7, 11, 13, 17, 19, 23, 29, 31}** are set as the **wheel**.   
       To understand it more, see the pattern in the above image that the numbers exhibit. The [LCM](https://www.geeksforgeeks.org/lcm-gq/) of the first three Prime Numbers is 30. The numbers(less than 30) which are ending with 7, 1, and 3 and are not a multiple of 2, 3, and 5 and are always [prime](https://www.geeksforgeeks.org/prime-numbers/)i.e **{7, 11, 13, 17, 19, 23, 29}**. Adding the no. 31 to this list and then if we add multiples of **30** to any of the numbers in the list, it gives us a [Prime Number](https://www.geeksforgeeks.org/prime-numbers/).

**Approach:**   
Following is the approach for the above algorithm: 

1. For [Primality Test](https://www.geeksforgeeks.org/primality-test-set-1-introduction-and-school-method/) of a given Number **N**, check if the given number is divisible by any of the number 2, 3, 5 or not.
2. If the number is not divisible by any of 2, 3, 5, then check if the number formed by adding multiples of 30 in the list [7, 11, 13, 17, 19, 23, 29, 31] divides the given number **N** or not. If Yes then the given number is not [Prime Number](https://www.geeksforgeeks.org/prime-numbers/), else it is a [Prime Number](https://www.geeksforgeeks.org/prime-numbers/).
   1. 
   2. Time Complexity: O(N3/2)
   3. Auxiliary Space: O(1)
3. Sieve in o(n) time – To Do