WEEK – 2 Research Work

Task 3: Write a function to calculate the greatest common divisor (GCD) and least common multiple (LCM) of two numbers.

GCD: Greatest Common Divisor is the largest number that divides all the given numbers.

- Used in number theory problems
- Cryptography problems

✅ 1. Greatest Common Divisor (GCD)

The **GCD** of two numbers is the largest number that divides both of them exactly (no remainder).

We use the **Euclidean Algorithm** to find the GCD:

🔁 Euclidean Algorithm Steps:

- If `b = 0`, then `GCD(a, b) = a`.
- Otherwise, repeatedly do:

```mathematica
GCD(a, b) = GCD(b, a % b)
```

- This process continues until the remainder is 0.

🔍 Example:

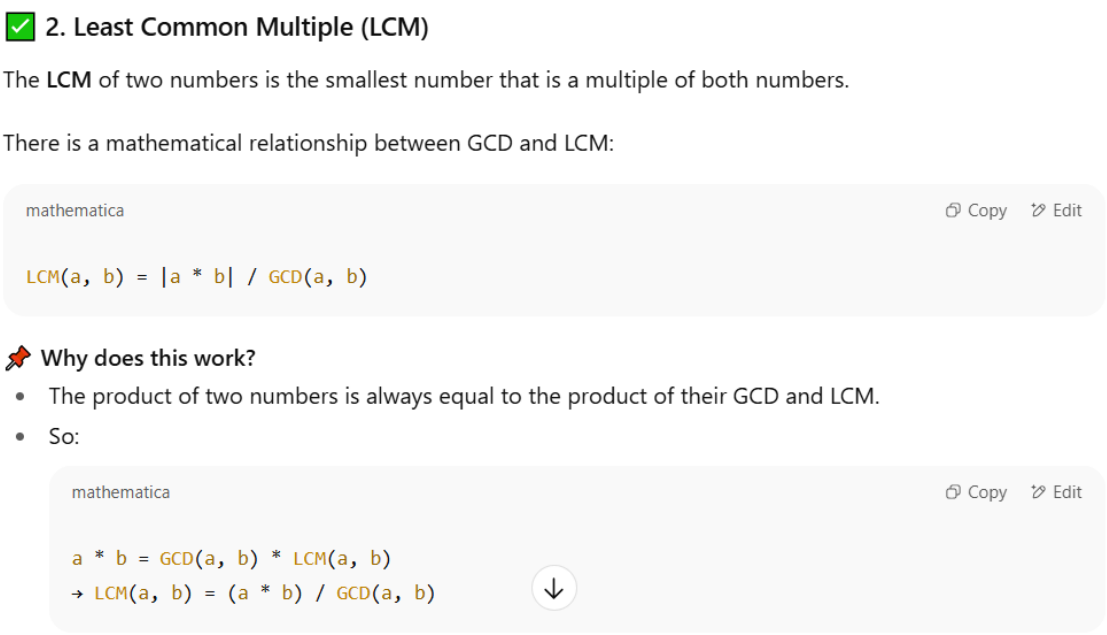
To find GCD of 12 and 18:

```mathematica
GCD(18, 12)
→ GCD(12, 18 % 12) = GCD(12, 6)
→ GCD(6, 12 % 6) = GCD(6, 0)
→ So, GCD = 6
```

LCM: Least Common Multiple is the smallest number which is the common multiple of all the given numbers.

- Used in time synchronization problems
- Event planning that occurs after equal time intervals

Task 4: Implement a program to find all the prime factors of a given number.



✅ 2. Least Common Multiple (LCM)

The **LCM** of two numbers is the smallest number that is a multiple of both numbers.

There is a mathematical relationship between GCD and LCM:

mathematica — Copy — Edit

```
LCM(a, b) = |a * b| / GCD(a, b)
```

📌 Why does this work?

- The product of two numbers is always equal to the product of their GCD and LCM.
- So:

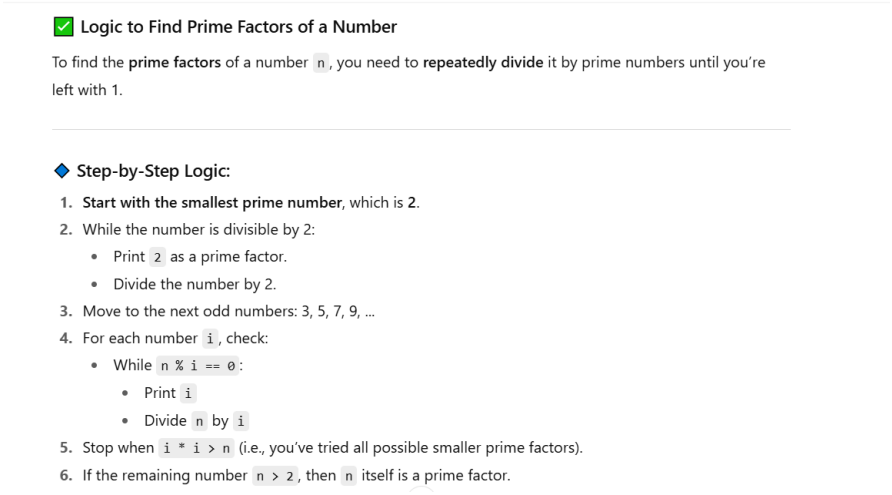
mathematica — Copy — Edit

```
a * b = GCD(a, b) * LCM(a, b)
→ LCM(a, b) = (a * b) / GCD(a, b)
```

Task 4: Implement a program to find all the prime factors of a given number.

Prime Factors: These are the prime numbers that are multiplied together to give the original number.

- Used in finding GCD and LCM
- Used in cryptography



✅ Logic to Find Prime Factors of a Number

To find the **prime factors** of a number `n`, you need to **repeatedly divide** it by prime numbers until you're left with 1.

🔷 Step-by-Step Logic:

1. **Start with the smallest prime number**, which is **2**.
2. While the number is divisible by 2:
   - Print `2` as a prime factor.
   - Divide the number by 2.
3. Move to the next odd numbers: 3, 5, 7, 9, ...
4. For each number `i`, check:
   - While `n % i == 0`:
     - Print `i`
     - Divide `n` by `i`
5. Stop when `i * i > n` (i.e., you've tried all possible smaller prime factors).
6. If the remaining number `n > 2`, then `n` itself is a prime factor.

Task 5: Write a program that takes a list of numbers and finds the subarray with the maximum sum (Kadane's Algorithm).

Kadane's Algorithm: Target is to find out contiguous numbers in a list that gives the maximum sum.

- Used in data compression
- Financial market analysis
- Image Processing

python

Copy Edit

```python
def max_subarray_sum(arr):
    max_so_far = arr[0]
    max_ending_here = arr[0]

    for num in arr[1:]:
        max_ending_here = max(num, max_ending_here + num)
        max_so_far = max(max_so_far, max_ending_here)

    return max_so_far

# Example usage:
nums = list(map(int, input("Enter numbers separated by space: ").split()))
print("Maximum subarray sum is:", max_subarray_sum(nums))
```

How this works (simple):

- It iterates over each number.
- Either starts fresh from current number or adds it to the current sum.
- Keeps track of the best (maximum) sum found so far.

I have also studied this topic from following YouTube video:

Link: https://youtu.be/9IZYqostl2M?si=a8pRsM1uR0VgXhgO