Project 4: Report

Exercise from Project 1:

Exercise 1

Design and implement your own algorithm that takes the array A with size m+n as input where:

- Subarray A[1], A[2],...A[m] sorted in ascending order
- Subarray A[m+1], A[m+2],...A[n] sorted in ascending order

and merges the two subarrays using an auxiliary array Aux of size min {m, n} back into array A sorted in ascending order. You must design and implement your own sorting function. Use of sorting functions in libraries is not permitted.

Test cases
Test Case 1: {} and {3, 7, 9}
Test Case 2: {2, 7, 9} and {1}
Test Case 3: {1, 7, 10, 15} and {3, 8, 12, 18}
Test Case 4: {1, 3, 5, 5, 15, 18, 21} and {5, 5, 6, 8, 10, 12, 16, 17, 17, 20, 25, 28}

Answer:

(1) The version of ChatGPT:
ChatGPT 3.5

(2) How many different ways of communication have you tried in the process and which one is the best?
8 different ways of communication and prompts have been given to chatGPT.

The best one is:

Design and implement an algorithm in java that takes the array A with size m+n as input where:
* Subarray A[1], A[2],...A[m] sorted in ascending order
* Subarray A[m+1], A[m+2],...A[n] sorted in ascending order
and merges the two subarrays using an auxiliary array Aux of size min {m, n} back into array A sorted in ascending order. You must design and implement your own sorting function. Use of sorting functions in libraries is not permitted.
Input array: 13572468
output: 12345678

Add a logic to swap the element's value after comparing. Try using pointers at the beginning of the subarrays. now merge the two subarrays using an auxiliary array Aux of size min {m, n} back into array A sorted in ascending order instead of swapping the elements directly.

(3) Is the "best" answer from the ChatGPT correct? Make a comparison between the answer you submitted in Project 1 with ChatGPT's best answer.

Yes, the best answer is correct.
Comparison table:

My Code	ChatGPT code
Accepts input interactively through the console.	Fixed input array in the code.
Uses conditional statements to handle edge cases like empty subarrays.	Employs arrays and pointers to manage merging without using conditional checks for empty subarrays.
Uses multiple loops to iterate through and merge the subarrays.	Uses a single loop to merge the subarrays.
utilizes multiple loops and conditional statements to merge the subarrays, managing edge cases like empty subarrays along the way.	code primarily employs pointers and arrays for merging, aiming for a more streamlined merging process without explicit conditional checks for empty arrays.
generally offers good performance for small datasets.	potentially faster for larger datasets as it doesn't require manual input during runtime.
Tends to consume less memory because it dynamically reads and processes input.	May consume more memory as it predefines arrays for input and processing, potentially using more memory for larger datasets.

(4) Other observations or interesting things discovered in this project.

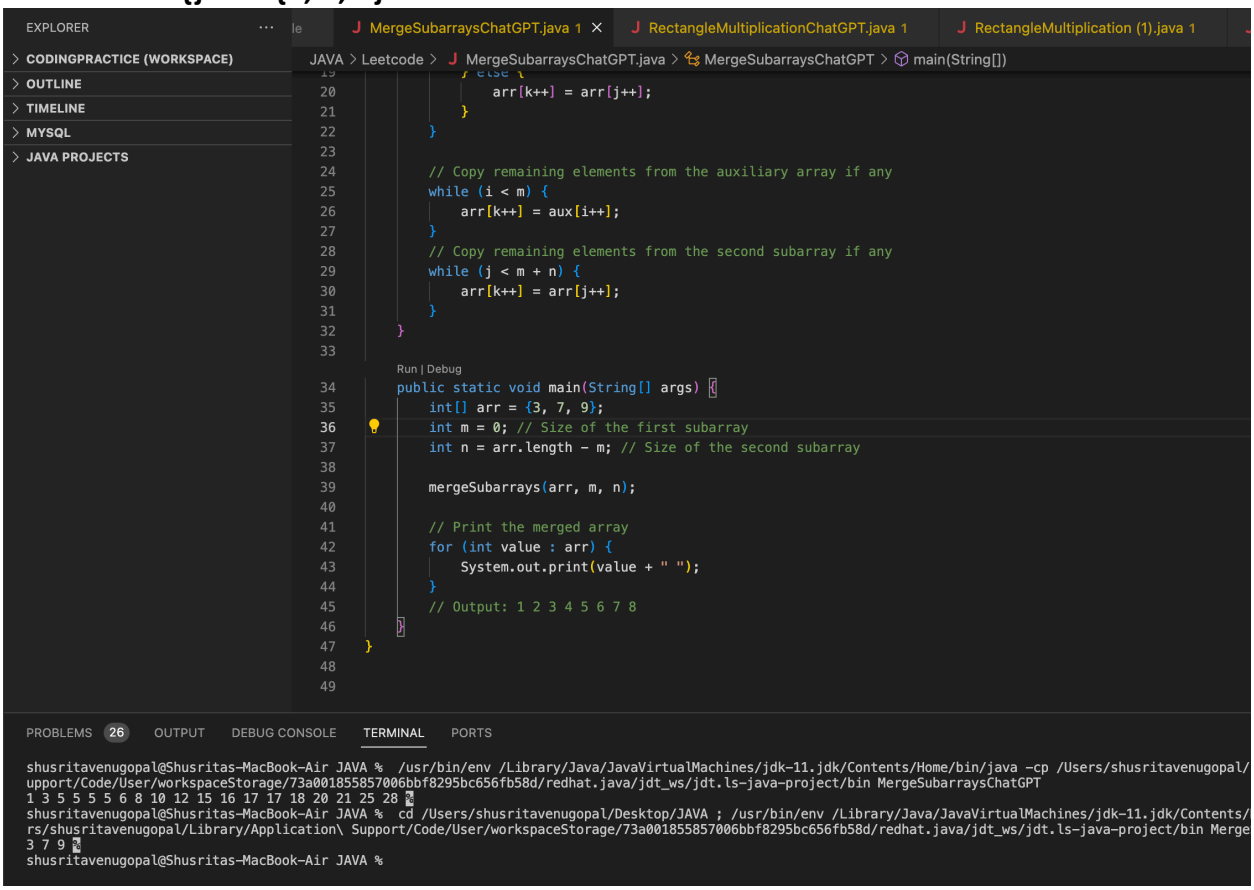
1. Problem Breakdown: Breaking down the problem statement into smaller parts with specific input and output examples helped in crafting a clearer and more accurate solution. This approach significantly improved the understanding of requirements.
2. Algorithmic Logic: The initial approach involved swapping elements directly and taking pointers at the end of subarrays, which didn't yield the expected result.

However, I asked to modify the logic to use pointers at the beginning of subarrays significantly improved the code's accuracy.

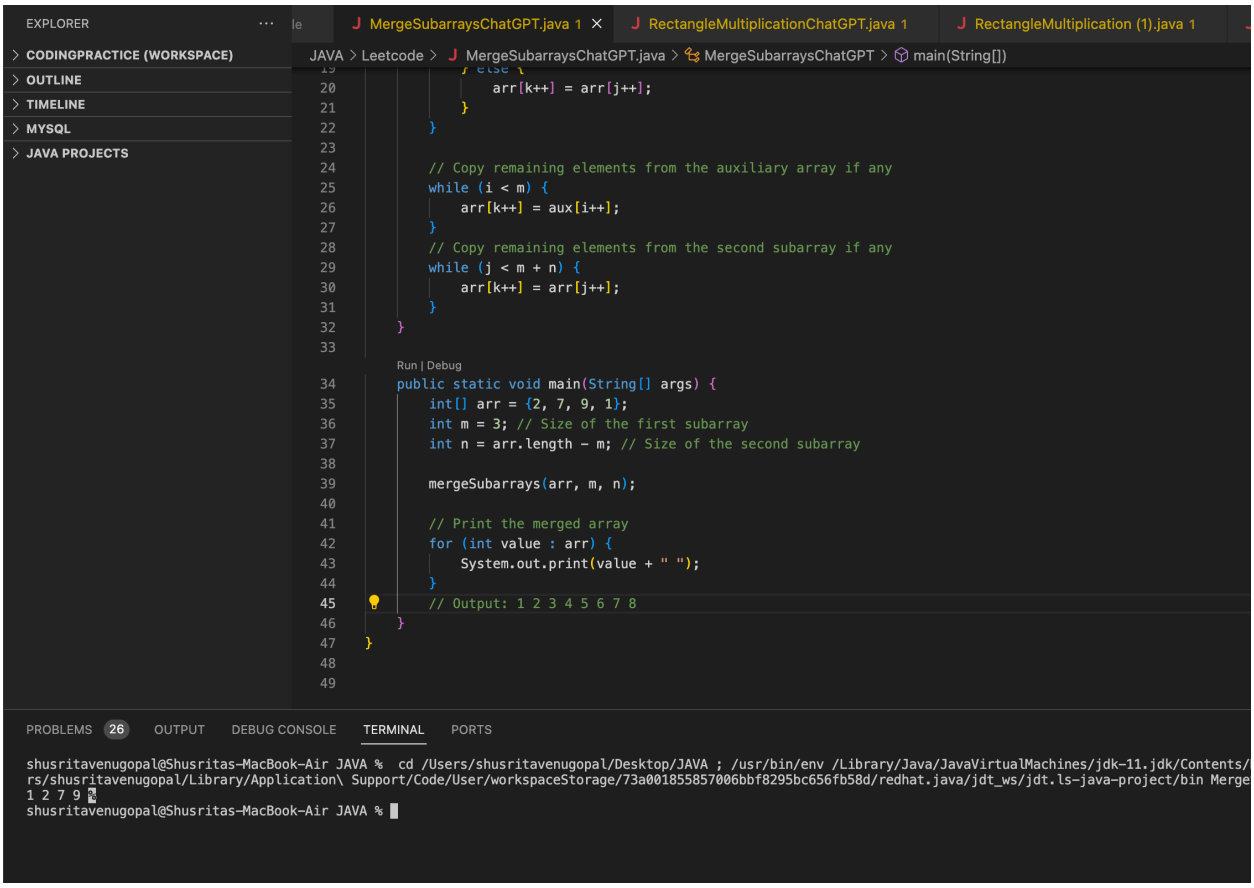
3. Collaborative Iteration: The iterative process involved multiple revisions, each addressing specific aspects. The collaborative approach of providing feedback and refining the solution gradually led to the correct implementation.
4. Error Identification: The process highlighted the importance of error identification and rectification, particularly in handling edge cases and boundary conditions.
5. Learning Through Mistakes: Mistakes and iterations throughout the process contributed to a deeper understanding of the problem statement and coding approaches, emphasizing learning from errors.
6. Algorithm Optimization: The final step involving the use of auxiliary arrays instead of direct element swapping demonstrated an optimized merging algorithm.

Output Screenshots:

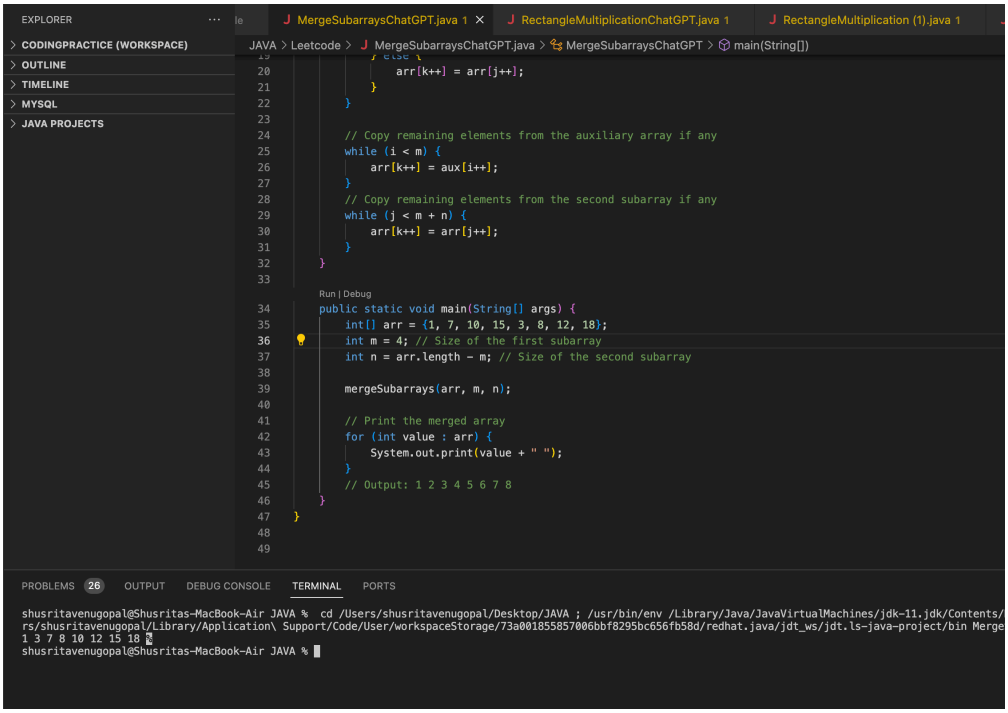
Test Case 1: {} and {3, 7, 9}



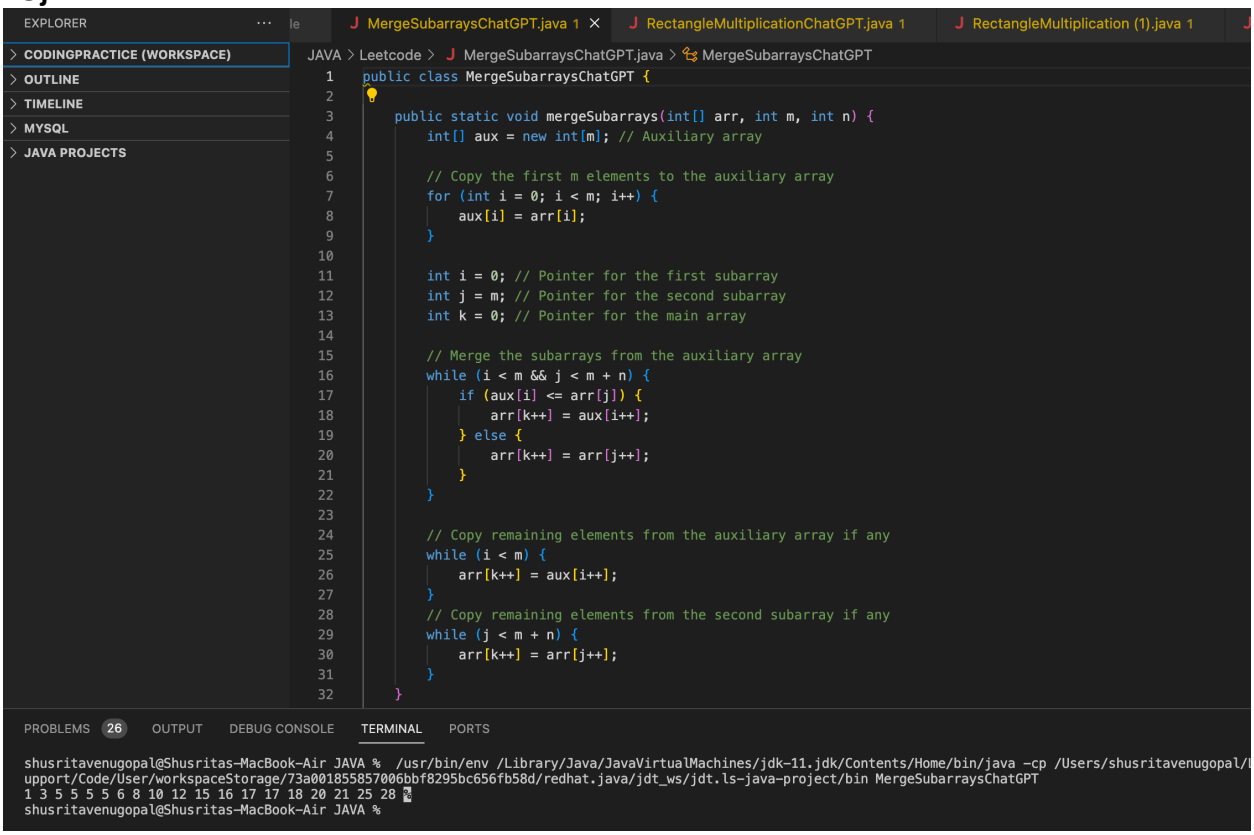
Test Case 2: {2, 7, 9} and {1}



Test Case 3: {1, 7, 10, 15} and {3, 8, 12, 18}:



Test Case 4: {1, 3, 5, 5, 15, 18, 21} and {5, 5, 6, 8, 10, 12, 16, 17, 17, 20, 25, 28}



Exercise 3

Design and implement your own algorithms, one for Ala Carte Multiplication and one for Rectangle Multiplication. Your algorithms must allow for both positive and negative multiplicands and multipliers.

Test cases

Test Case 1: 7000 * 7294 Test Case 2: 25 * 5038385 Test Case 3: -59724 * 783 Test Case 4: 8516 * -82147953548159344 Test Case 5: 45952456856498465985 * 98654651986546519856

Test Case 6: -45952456856498465985 * -98654651986546519856

Alacarte:

(1) The version of ChatGPT:

ChatGPT 3.5

(2) How many different ways of communication have you tried in the process and which one is the best?

3 ways of communication and different prompts were used to achieve correct answer.

The best one is:

Design and implement your own algorithm in python for Ala Carte Multiplication Your algorithms must allow for both positive and negative multiplicands and multipliers.
Test cases
Test Case 1: 7000 * 7294

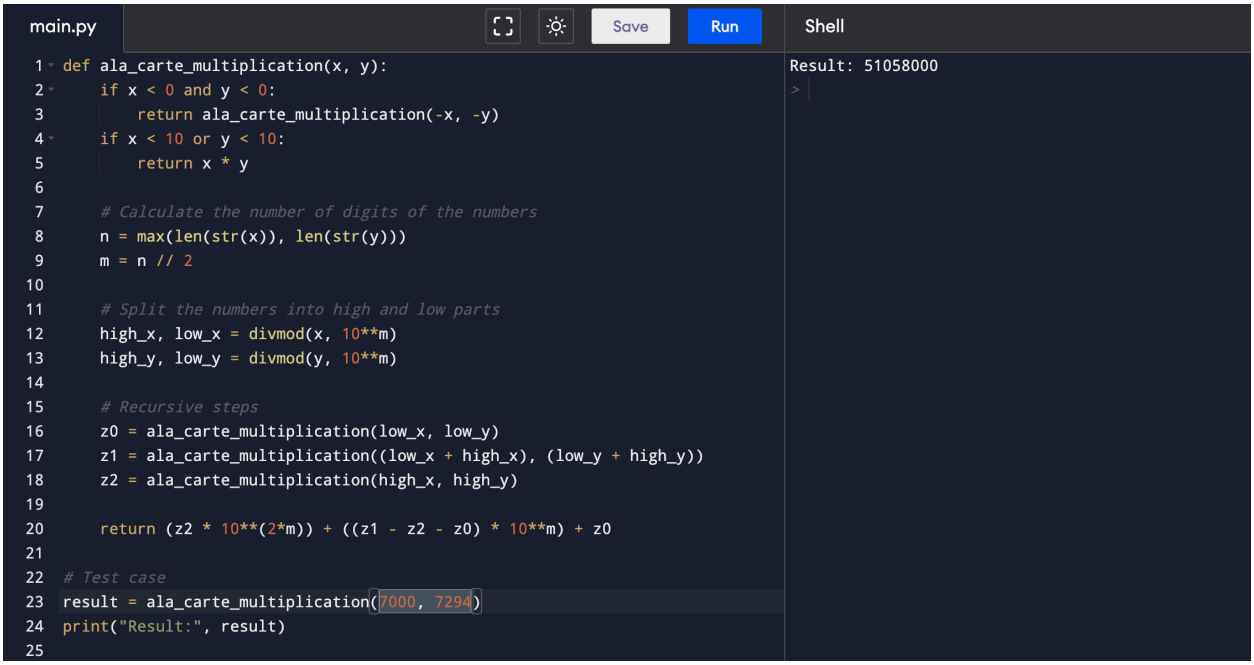
Ala carte multiplication means A multiplication algorithm is a method for multiplying two numbers. The efficiency of an algorithm depends on the size of the numbers being multiplied.

Karatsuba is one of the multiplication algorithm that uses Ala carte multiplication. It is a divide-and-conquer algorithm that reduces the multiplication of two n-digit numbers to three multiplications of n/2-digit numbers and, by repeating this reduction, to at most {\displaystyle n^{\log _{2}3}\approx n^{1.58}} single-digit multiplications. It is therefore asymptotically faster than the traditional algorithm, which performs
single-digit products.

You need to use karatsuba approach to implement this algorithm.

(3) Is the "best" answer from the ChatGPT correct? Make a comparison between the answer you submitted in Project 1 with ChatGPT's best answer.

Yes, best answer is correct. Attaching screenshots for all test case:



main.py

Save Run

```python
def ala_carte_multiplication(x, y):
    if x < 0 and y < 0:
        return ala_carte_multiplication(-x, -y)
    if x < 10 or y < 10:
        return x * y

    # Calculate the number of digits of the numbers
    n = max(len(str(x)), len(str(y)))
    m = n // 2

    # Split the numbers into high and low parts
    high_x, low_x = divmod(x, 10**m)
    high_y, low_y = divmod(y, 10**m)

    # Recursive steps
    z0 = ala_carte_multiplication(low_x, low_y)
    z1 = ala_carte_multiplication((low_x + high_x), (low_y + high_y))
    z2 = ala_carte_multiplication(high_x, high_y)

    return (z2 * 10**(2*m)) + ((z1 - z2 - z0) * 10**m) + z0

# Test case
result = ala_carte_multiplication(25, 5038385)
print("Result:", result)

```

Shell

```
Result: 125959625
>
```

main.py

Save Run

```python
def ala_carte_multiplication(x, y):
    if x < 0 and y < 0:
        return ala_carte_multiplication(-x, -y)
    if x < 10 or y < 10:
        return x * y

    # Calculate the number of digits of the numbers
    n = max(len(str(x)), len(str(y)))
    m = n // 2

    # Split the numbers into high and low parts
    high_x, low_x = divmod(x, 10**m)
    high_y, low_y = divmod(y, 10**m)

    # Recursive steps
    z0 = ala_carte_multiplication(low_x, low_y)
    z1 = ala_carte_multiplication((low_x + high_x), (low_y + high_y))
    z2 = ala_carte_multiplication(high_x, high_y)

    return (z2 * 10**(2*m)) + ((z1 - z2 - z0) * 10**m) + z0

# Test case
result = ala_carte_multiplication(8516, -82147953548159344)
print("Result:", result)

```

Shell

```
Result: -699571972416124973504
>
```

main.py

Save Run

```python
def ala_carte_multiplication(x, y):
    if x < 0 and y < 0:
        return ala_carte_multiplication(-x, -y)
    if x < 10 or y < 10:
        return x * y

    # Calculate the number of digits of the numbers
    n = max(len(str(x)), len(str(y)))
    m = n // 2

    # Split the numbers into high and low parts
    high_x, low_x = divmod(x, 10**m)
    high_y, low_y = divmod(y, 10**m)

    # Recursive steps
    z0 = ala_carte_multiplication(low_x, low_y)
    z1 = ala_carte_multiplication((low_x + high_x), (low_y + high_y))
    z2 = ala_carte_multiplication(high_x, high_y)

    return (z2 * 10**(2*m)) + ((z1 - z2 - z0) * 10**m) + z0

# Test case
result = ala_carte_multiplication(-45952456856498465985,
    -98654651986546519856)
print("Result:", result)
```

Shell

```
Result: 45334236391046496343970934505043430981600
>
```

main.py

Save Run

```python
def ala_carte_multiplication(x, y):
    if x < 0 and y < 0:
        return ala_carte_multiplication(-x, -y)
    if x < 10 or y < 10:
        return x * y

    # Calculate the number of digits of the numbers
    n = max(len(str(x)), len(str(y)))
    m = n // 2

    # Split the numbers into high and low parts
    high_x, low_x = divmod(x, 10**m)
    high_y, low_y = divmod(y, 10**m)

    # Recursive steps
    z0 = ala_carte_multiplication(low_x, low_y)
    z1 = ala_carte_multiplication((low_x + high_x), (low_y + high_y))
    z2 = ala_carte_multiplication(high_x, high_y)

    return (z2 * 10**(2*m)) + ((z1 - z2 - z0) * 10**m) + z0

# Test case
result = ala_carte_multiplication(45952456856498465985, 98654651986546519856)
print("Result:", result)

```

Shell

```
Result: 45334236391046496343970934505043430981600
>
```

Comparison:

My code	ChatGPT
Performs multiplication using bitwise operations and integer arithmetic.	Utilizes a recursive approach based on splitting the numbers into high and low parts.
Handles both positive and negative numbers by checking the sign with flags (`f1` and `f2`).	Handles both positive and negative numbers by considering absolute values.
focuses on bit manipulation and integer arithmetic to perform the multiplication operation.	divides the numbers into high and low parts, performing recursive multiplication and using mathematical formulae to merge the results

employs flags (f1 and f2) to determine the sign of the numbers and prints the result accordingly.	converts negative numbers to their absolute values and performs the multiplication on the positive equivalents.
uses a loop structure with bitwise operations to perform the multiplication in a stepwise manner.	involves a more structured approach, leveraging recursion to break down the numbers and calculate the final product.

(4) Other observations or interesting things discovered in this project.

1. Understanding Problem Context: Explaining the nature of the Karatsuba method helped in guiding towards the most efficient solution. Often, understanding the problem's context and background aids in finding more accurate and optimized solutions.

2. Iterative Learning Process: The multiple revisions showcased the iterative learning process. Each revision involved improvements, learnings from previous attempts, and ultimately led to a refined and correct solution. It underscores the significance of persistence and learning from mistakes in problem-solving.

3. Explanation Aids Understanding: Providing context and explaining the underlying logic helped in reaching a more accurate solution. Breaking down complex concepts into simpler explanations aids in better comprehension and problem-solving.

4. Optimal Solutions Align with Problem Nature: Optimal solutions often depend on the problem's nature and requirements. While there might be multiple solutions, the one that aligns most closely with the problem specifications tends to be the most efficient and effective.

EXERCISE 3**:**

3. Design and implement your own algorithms, one for Rectangle Multiplication. Your algorithms must allow for both positive and negative multiplicands and multipliers.
Test cases
Test Case 1: 7000 * 7294 Test Case 2: 25 * 5038385 Test Case 3: -59724 * 783 Test Case 4: 8516 * -82147953548159344 Test Case 5: 45952456856498465985 * 98654651986546519856
Test Case 6: -45952456856498465985 * -98654651986546519856

(1) The version of ChatGPT.
ChatGPT 3

(2) **How many different ways of communication have you tried in the process and which one is the best?**
18 TIMES.

The best prompt for this algorithm is:
Design and implement your own algorithm in java for rectangle Multiplication
Your algorithms must allow for both positive and negative multiplicands and multipliers.
Test cases
Test Case 1: 7000 * 7294
In math, rectangle multiplication is a diagram used to solve multiplication problems. It's also known as the "Area model for multiplication" Implement the algorithm this way.
Can you implement rectangle multiplication method using big integers? HANDLE NEGATIVE AND POSITIVE NUMBERS.

(3) Is the "best" answer from the ChatGPT correct? Make a comparison between the answer you submitted in Project 1 with ChatGPT's best answer.
Yes the best answer form ChatGPT is correct.
Comparison:

My code	ChatGPT code
Uses an integer array for storing intermediate multiplication results, which takes additional space, specifically O(m + n), where m and n are the lengths of the input strings.	Primarily uses integer variables, with minimal extra space required. Space complexity is negligible, approximately O(1).
Uses strings for processing, providing flexibility in handling arbitrarily large numbers. However, string manipulations might affect the performance for extremely large inputs due to the nested loops.	Deals with integer manipulations directly, potentially more efficient for smaller to medium-sized inputs, but it's less adaptable for very large inputs due to integer overflow concerns.
Utilizes a string-based approach, performing digit-wise multiplication and managing signs separately.	Implements a numerical approach, handling multiplication by iterating over digits and keeping track of intermediate results.
Employs a more abstract approach with string operations, might be complex to follow due to multiple loops and handling signs.	Utilizes more direct mathematical operations, focusing on the multiplication algorithm's essence.
Separates the multiplication logic into a function rectangleMultiplication,	Embeds the multiplication logic within the main function, potentially

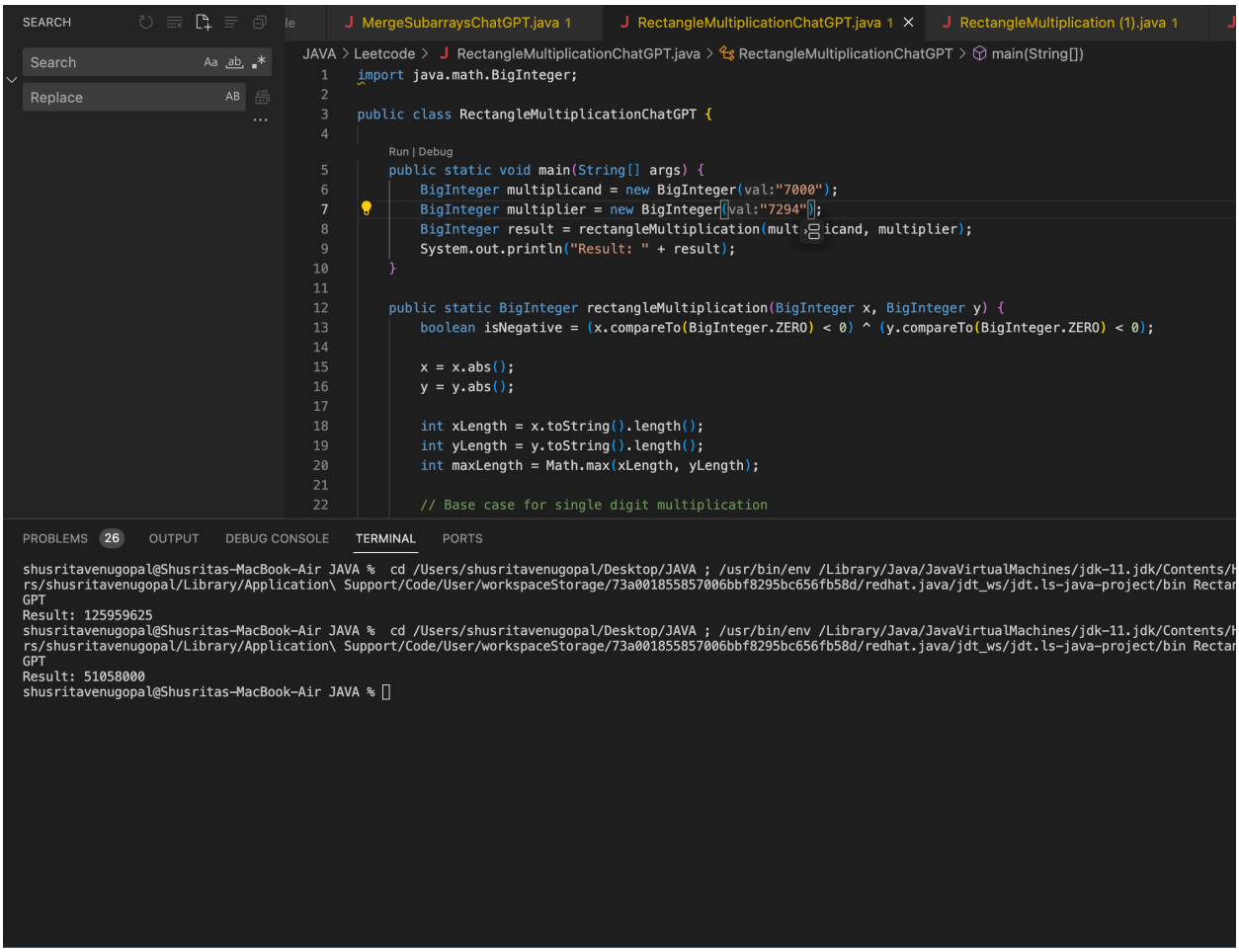
enhancing code modularity and reusability.	reducing modularity for reuse in other parts of the codebase.

(4) Other observations or interesting things discovered in this project.

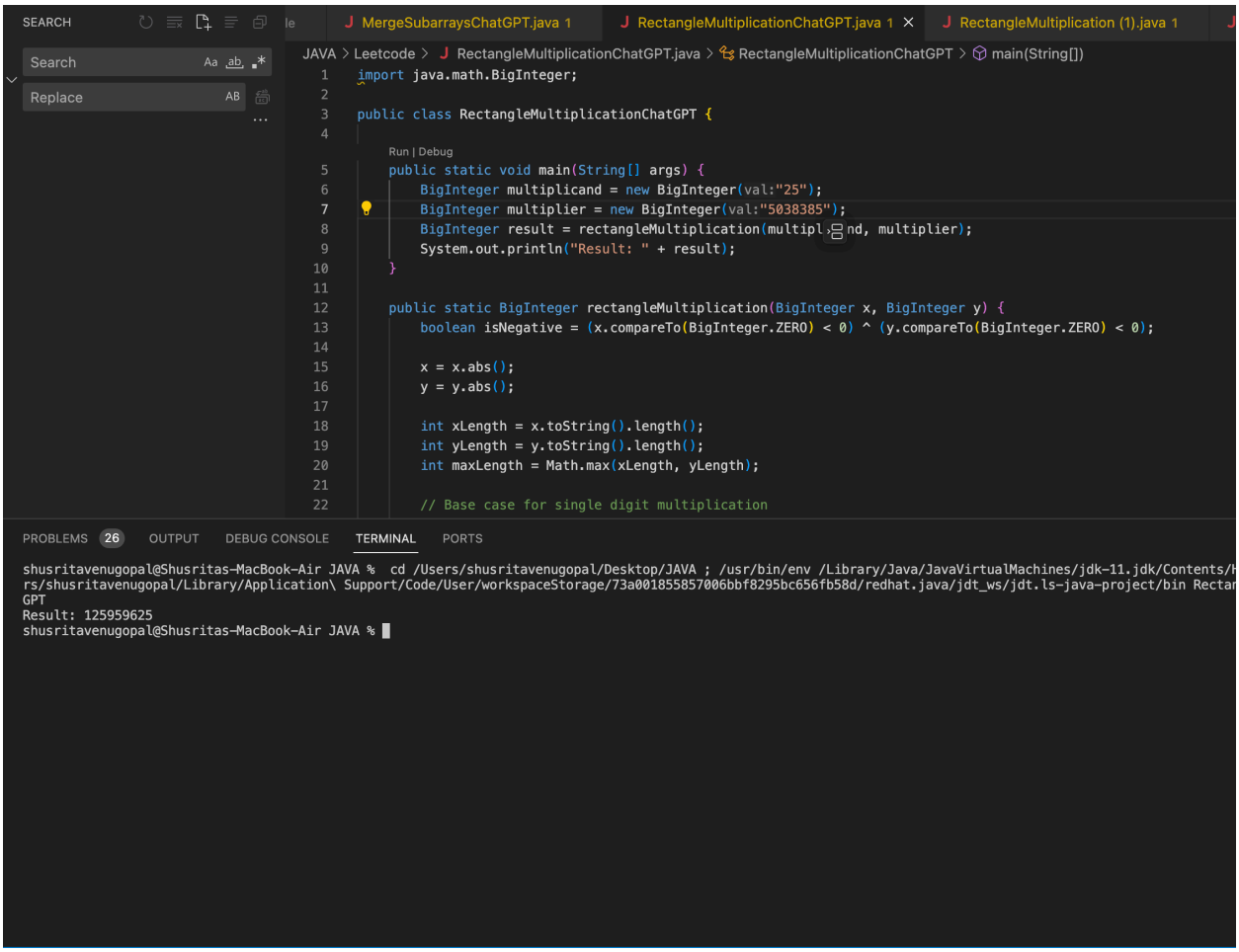
1. Understanding the Context is Crucial: Providing context or clarifying the mathematical concept aligned the solution better with the intended objective.
2. Domain Knowledge Enhances Solution: Demonstrating a conceptual understanding of mathematical operations (like rectangle multiplication) guided the coding approach more effectively.
3. Adaptation and Flexibility: Adapting the approach based on explanations and examples illustrated the importance of flexibility in problem-solving.
4. Application-Oriented Solutions: Translating theoretical concepts into functional code demonstrated the practical application of mathematical ideas.
5. Collaborative Problem-Solving: The collaborative process between the requester and the responder led to a more accurate and aligned solution.
6. Shared Understanding: Clear communication and shared understanding were pivotal in achieving the desired outcome.

Output screenshots:

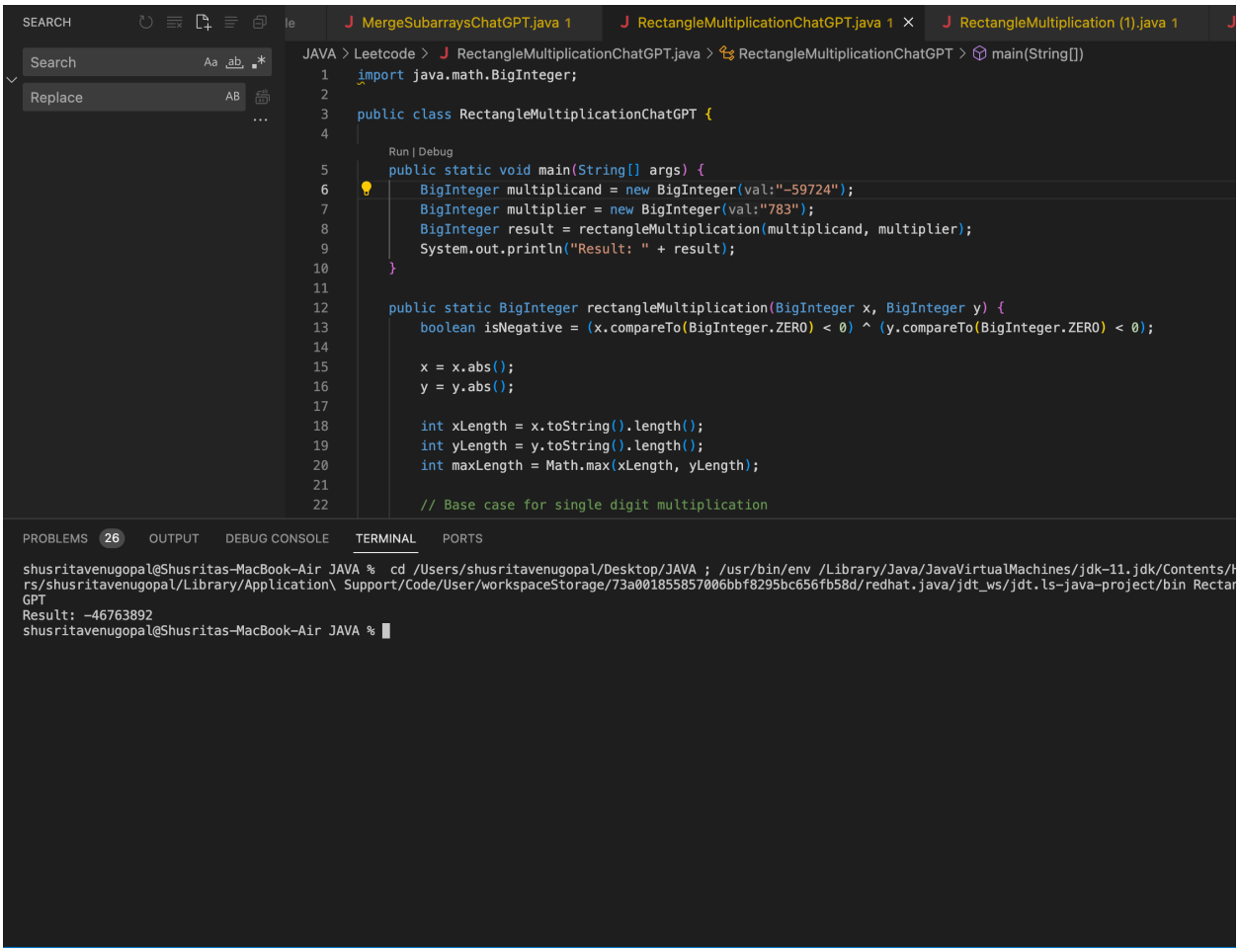
Test Case 1: 7000 * 7294



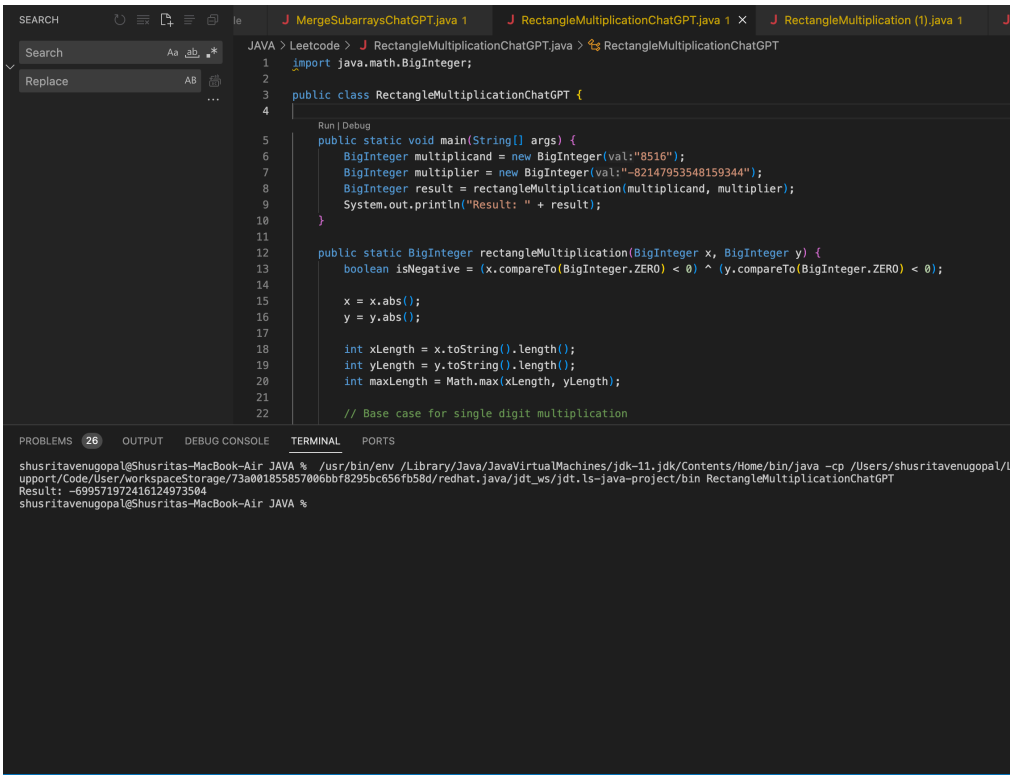
Test Case 2: 25 * 5038385



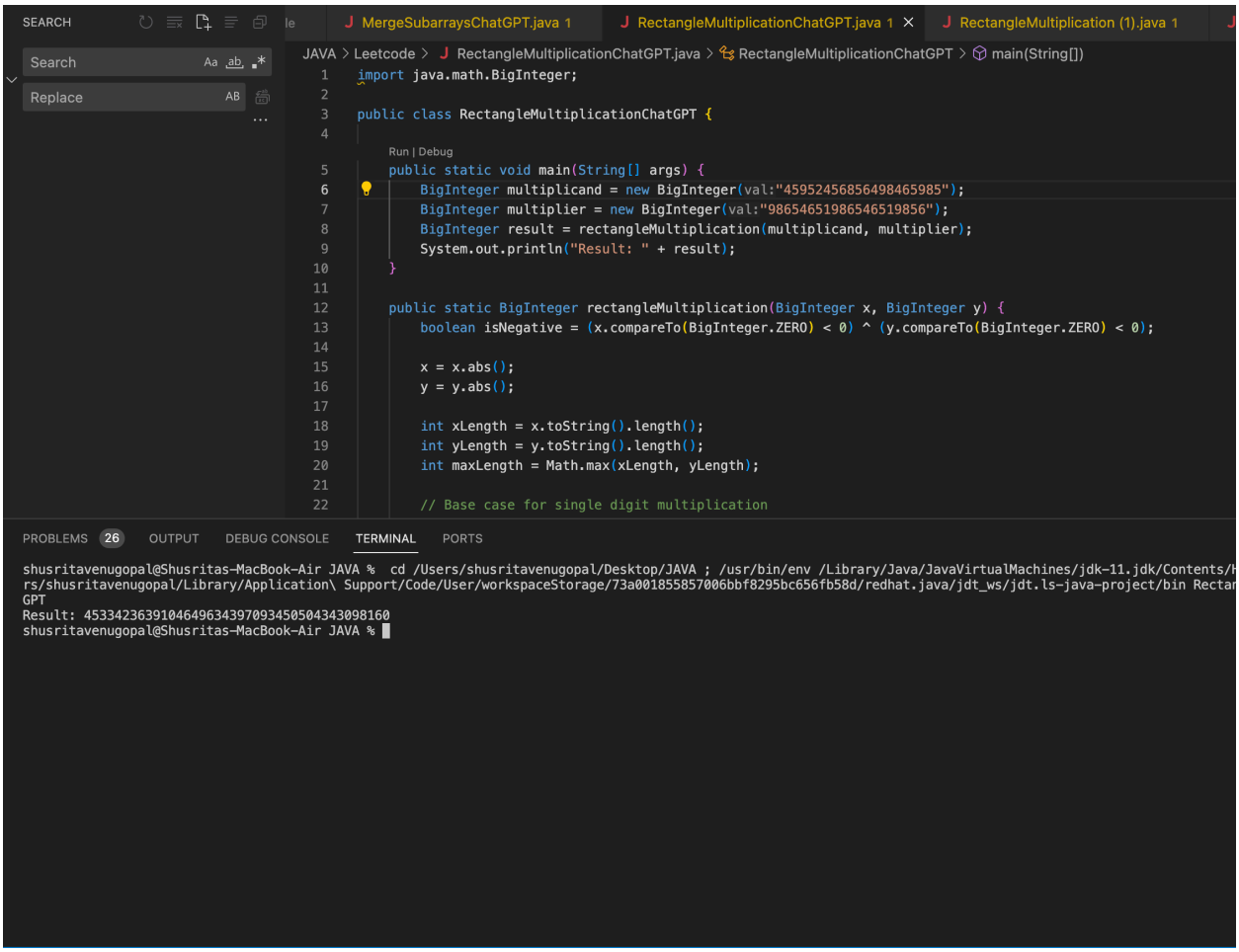
Test Case 3: -59724 * 783



Test Case 4: 8516 * -82147953548159344



Test Case 5: 45952456856498465985 * 98654651986546519856



Test Case 6: -45952456856498465985 * -98654651986546519856