**DAA Assignment - 04**

**Name :Gayatri Avhad**

**Reg. no. :** 231071007

**Branch :** Computer Engineering

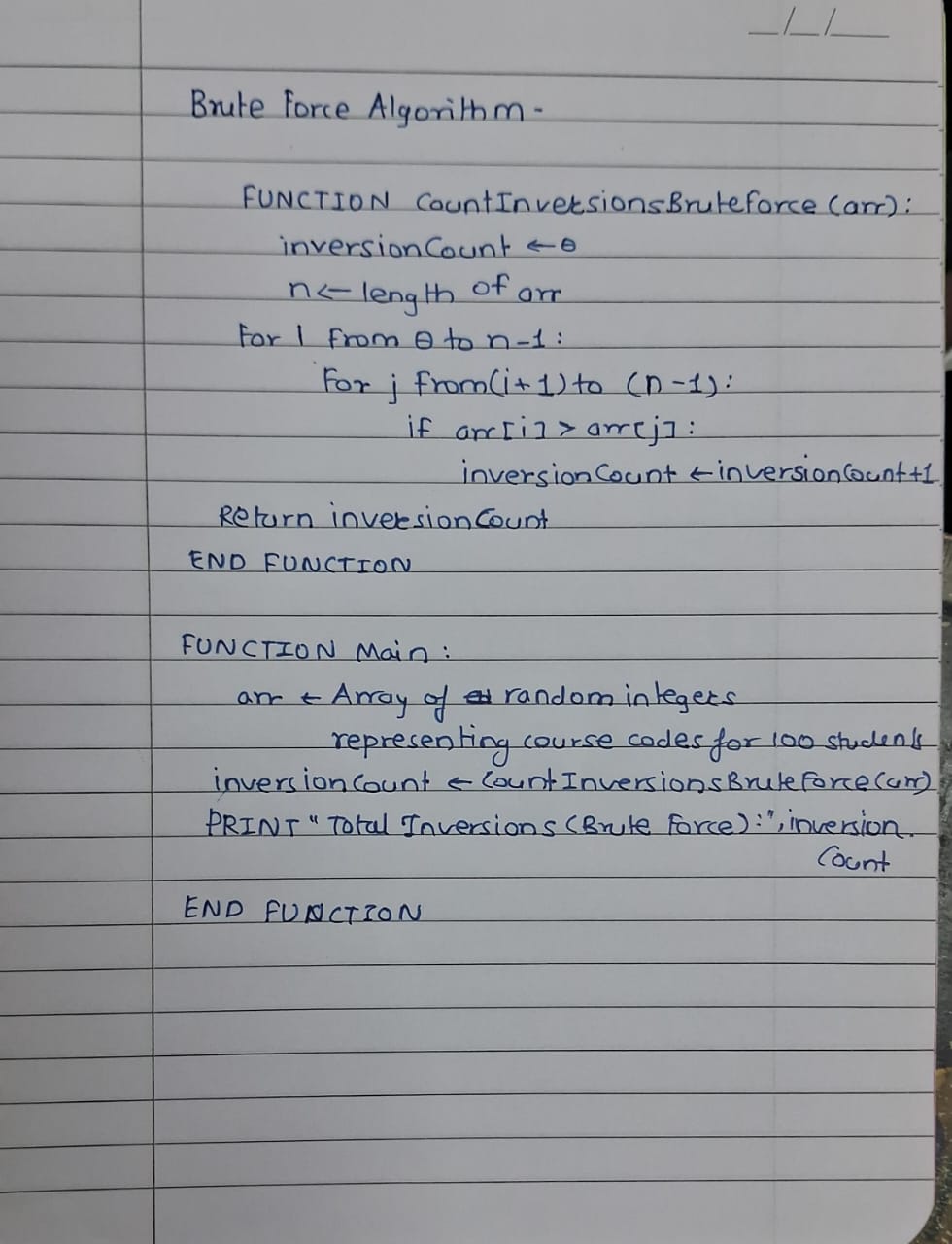
**Batch A**

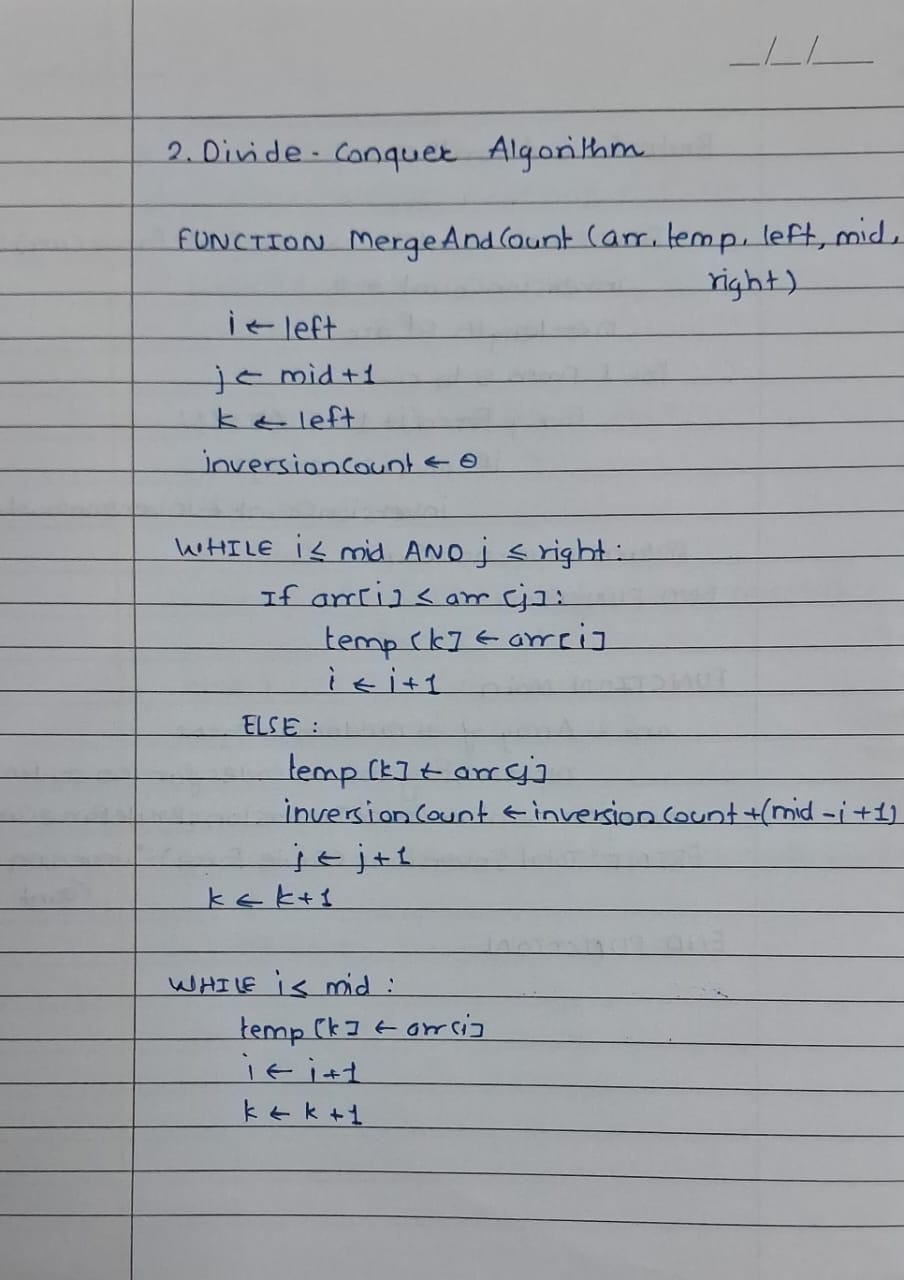
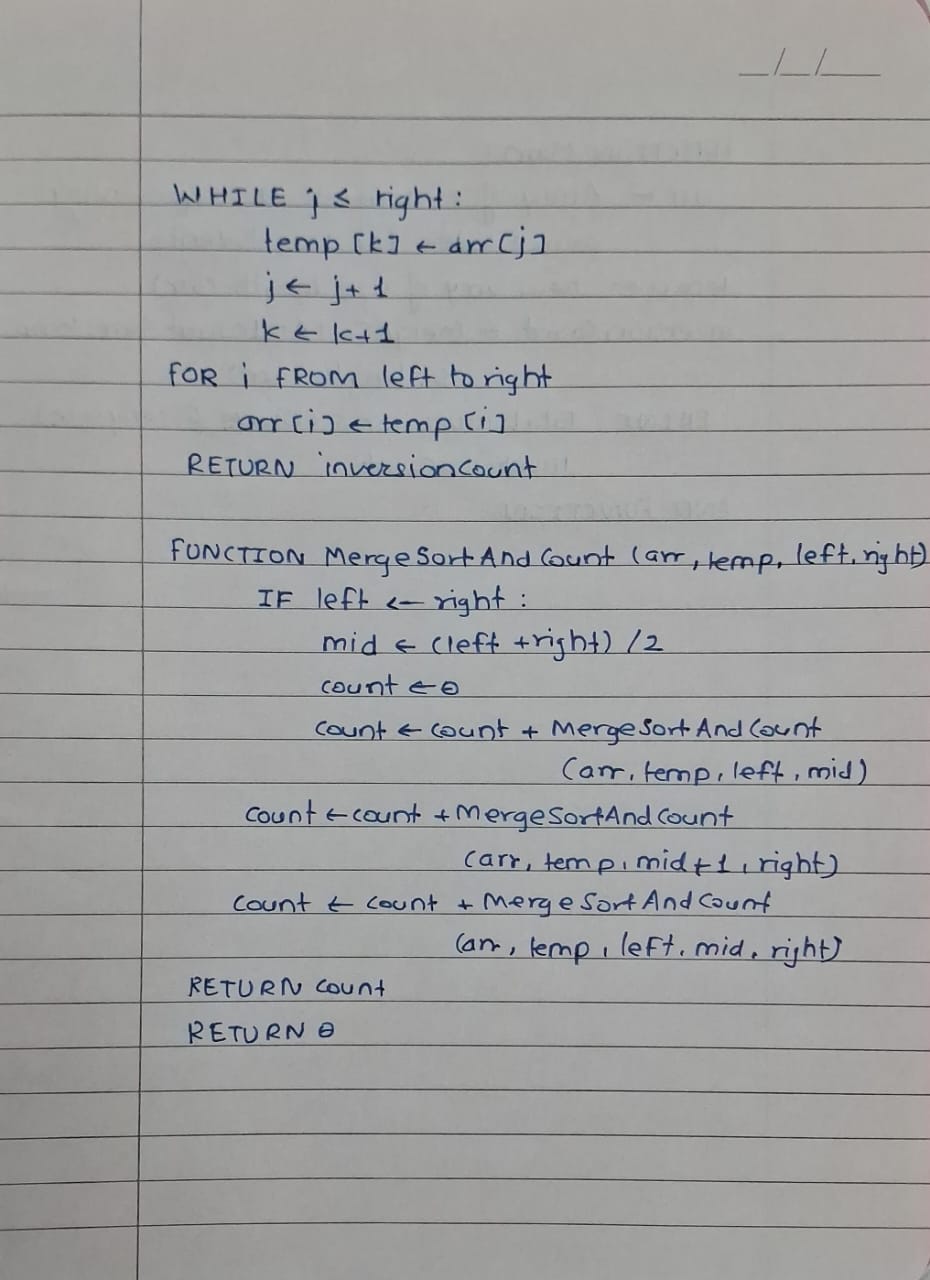
**Aim 1 :** Consider first/second year course-code choices of 100 students.

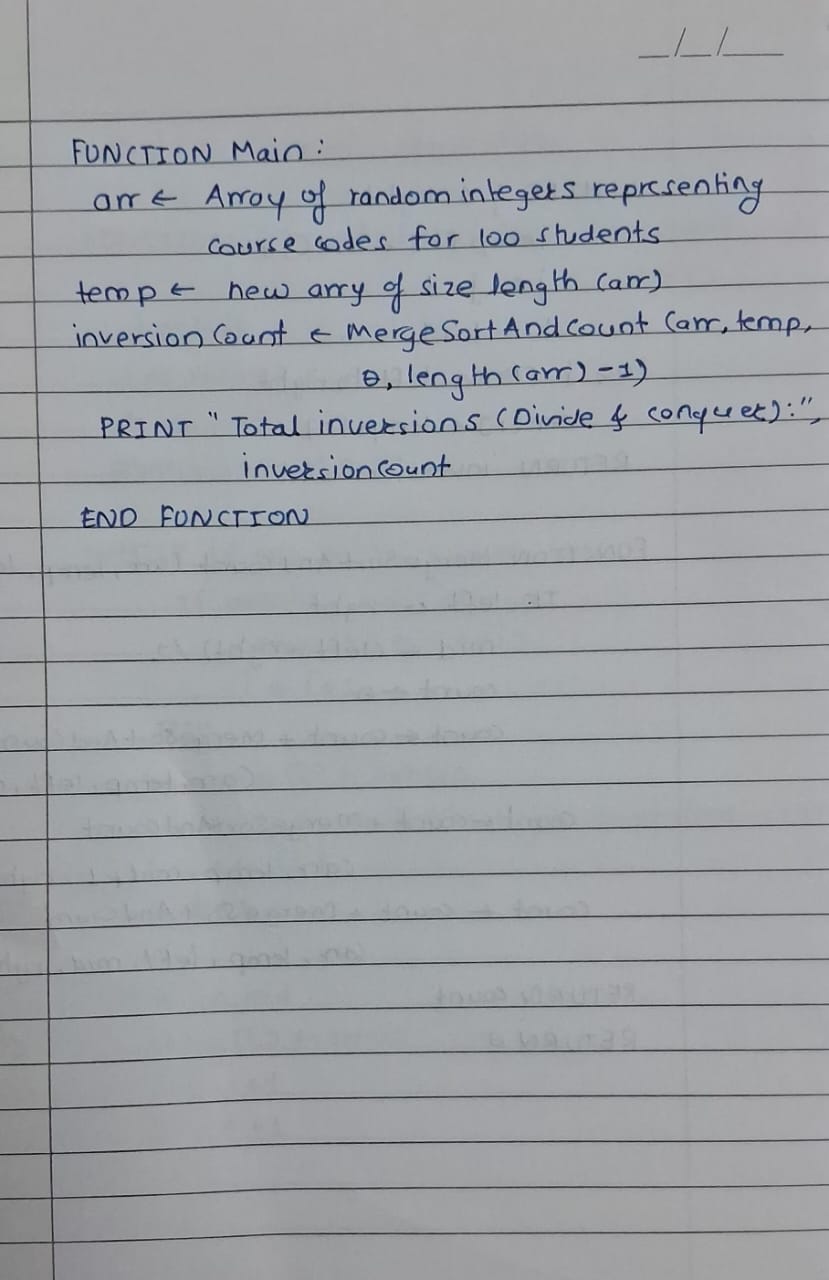
Find the inversion count of these choices.

Find students with zero, one, two, three inversion counts comment on your result.

**Algorithm :**







**Test Case 1** (Positive):

**Input:**  
Student 1: [1, 2, 3, 4]

Student 2: [4, 3, 2, 1]

Student 3: [2, 1, 4, 3]

Student 4: [1, 3, 2, 4]

**Output:**  
Student 1 Inversion Count: 0

Student 2 Inversion Count: 6

Student 3 Inversion Count: 3

Student 4 Inversion Count: 1

Number of students with 0 inversions: 1

Number of students with 1 inversion: 1

Number of students with 2 inversions: 0

Number of students with 3 or more inversions: 2

**Test Cases** (Negative):

**Input:**Student 1: [3, 2, -1, 4] // Invalid negative course code

**Output:**Error: Invalid course code detected. Please enter positive course codes only.

**Input:**Student 1: [1, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,4] // Exceeds maximum number of allowed courses

**Output:**Error: Exceeds maximum course limit. Maximum of 10 course codes allowed per student.

**Input:**  
Student 1: [] // Empty list of course codes

**Output:**  
Error: No course codes entered. Each student must select at least one course.

**Input:**  
Student 1: [1, 2, 'X', 3] // Invalid character in course code list

**Output:**  
Error: Invalid input. All course codes must be numeric values.

**Input:**  
Student 1: [1, 1, 1, 1, 1] // Duplicate course codes

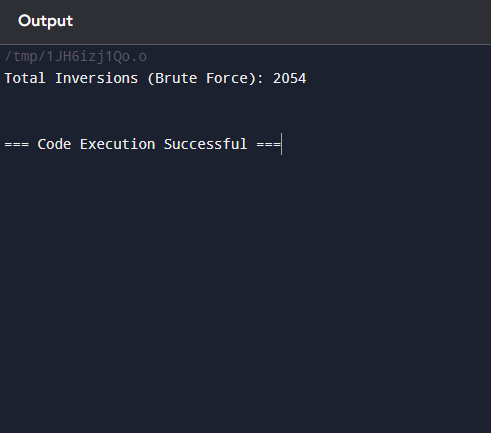
**Output:**  
Error: Duplicate course codes found. Each student must select unique course codes.

**1. Brute-force Multiplication Pseudocode:**

* **Input -**

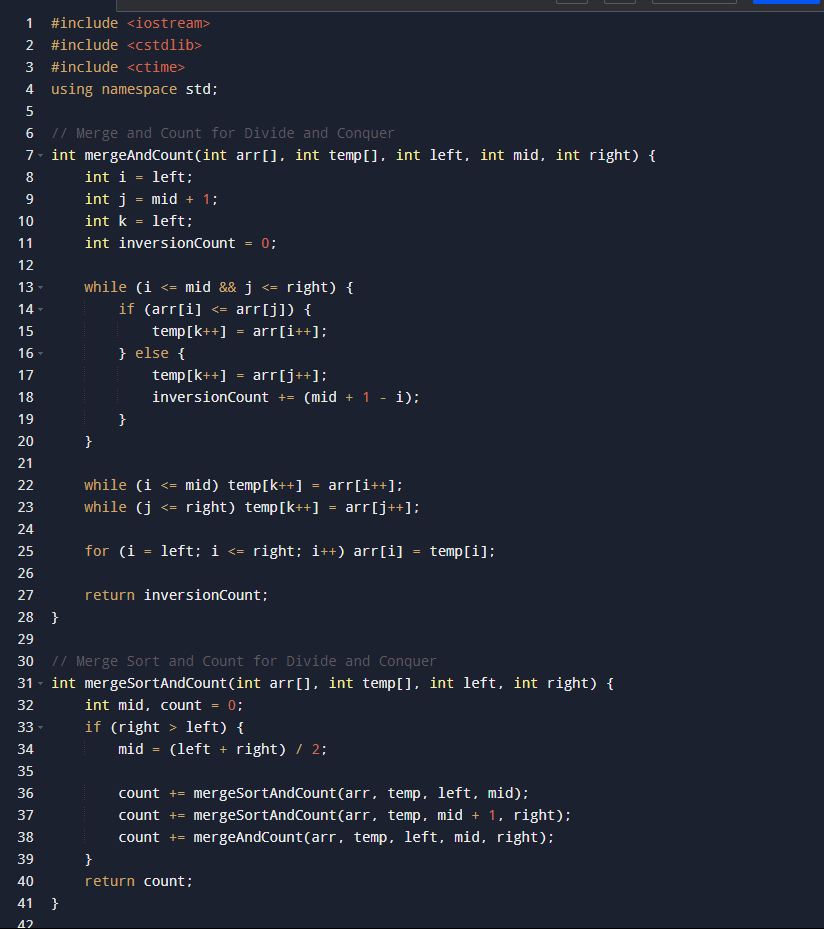
****

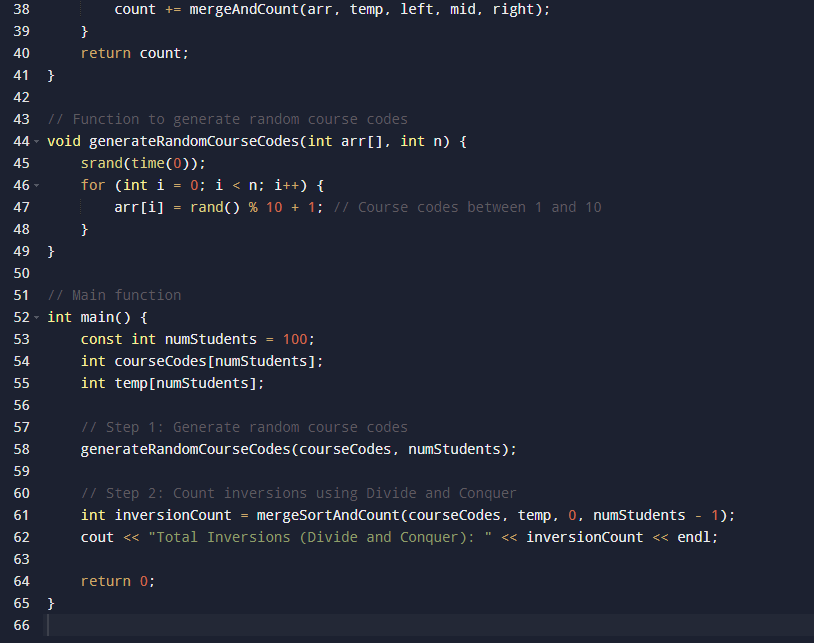
* **Output -**

****

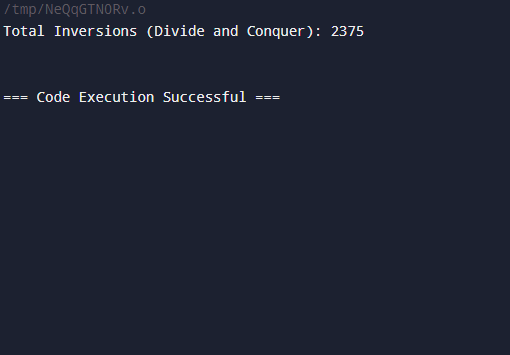
**2. Divide-and-Conquer Multiplication (Karatsuba) Pseudocode:**

* **Input -**

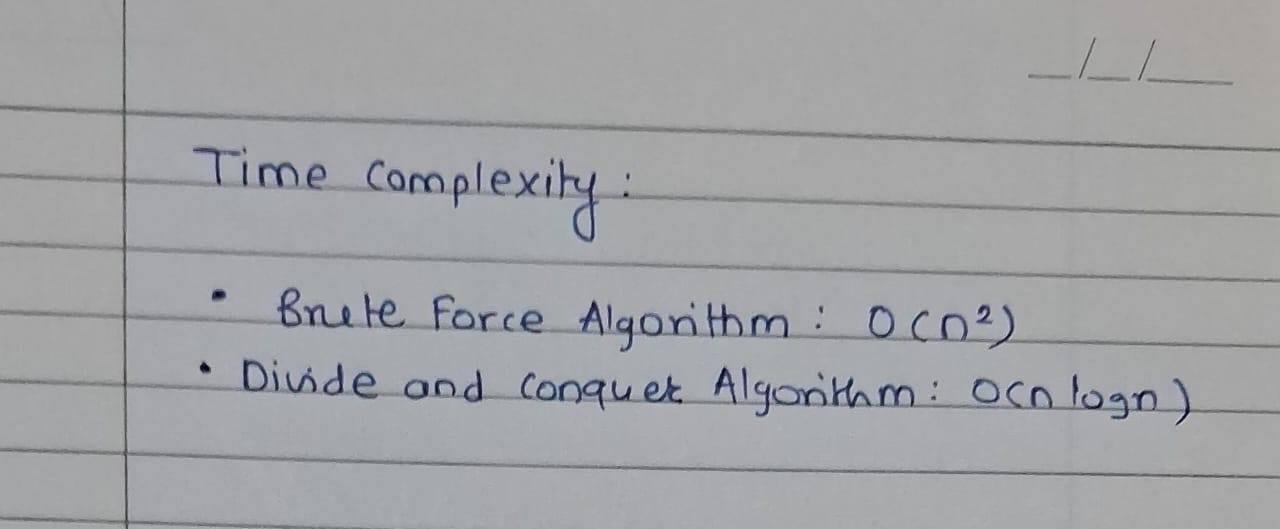
****

****

* **Output -**

****

### **Time Complexity :**



**Conclusion :**

This experiment compared two algorithms for counting inversions in course-code choices among 100 students:

* The Brute Force Algorithm has a time complexity of O(n2)O(n^2)O(n2), while the Divide and Conquer Algorithm is more efficient at O(nlog⁡n)O(n \log n)O(nlogn).
* Both methods produced the same inversion count, confirming their accuracy.

In summary, the Divide and Conquer approach is preferred for its efficiency in handling larger datasets.

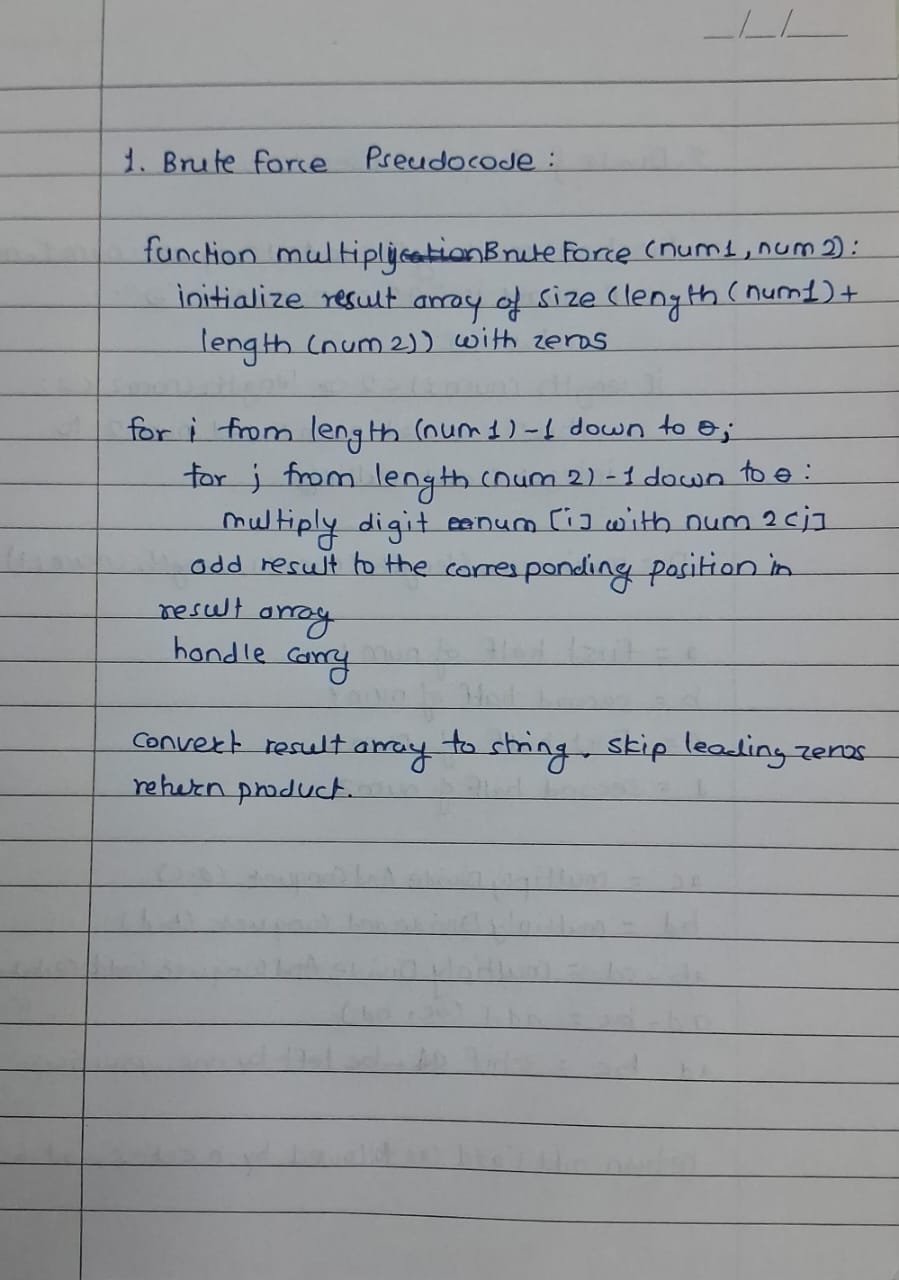
**Aim 2 :** Consider large integers of size 10, 50, 100, 500 and 1000 digits.

Write integer multiplication program

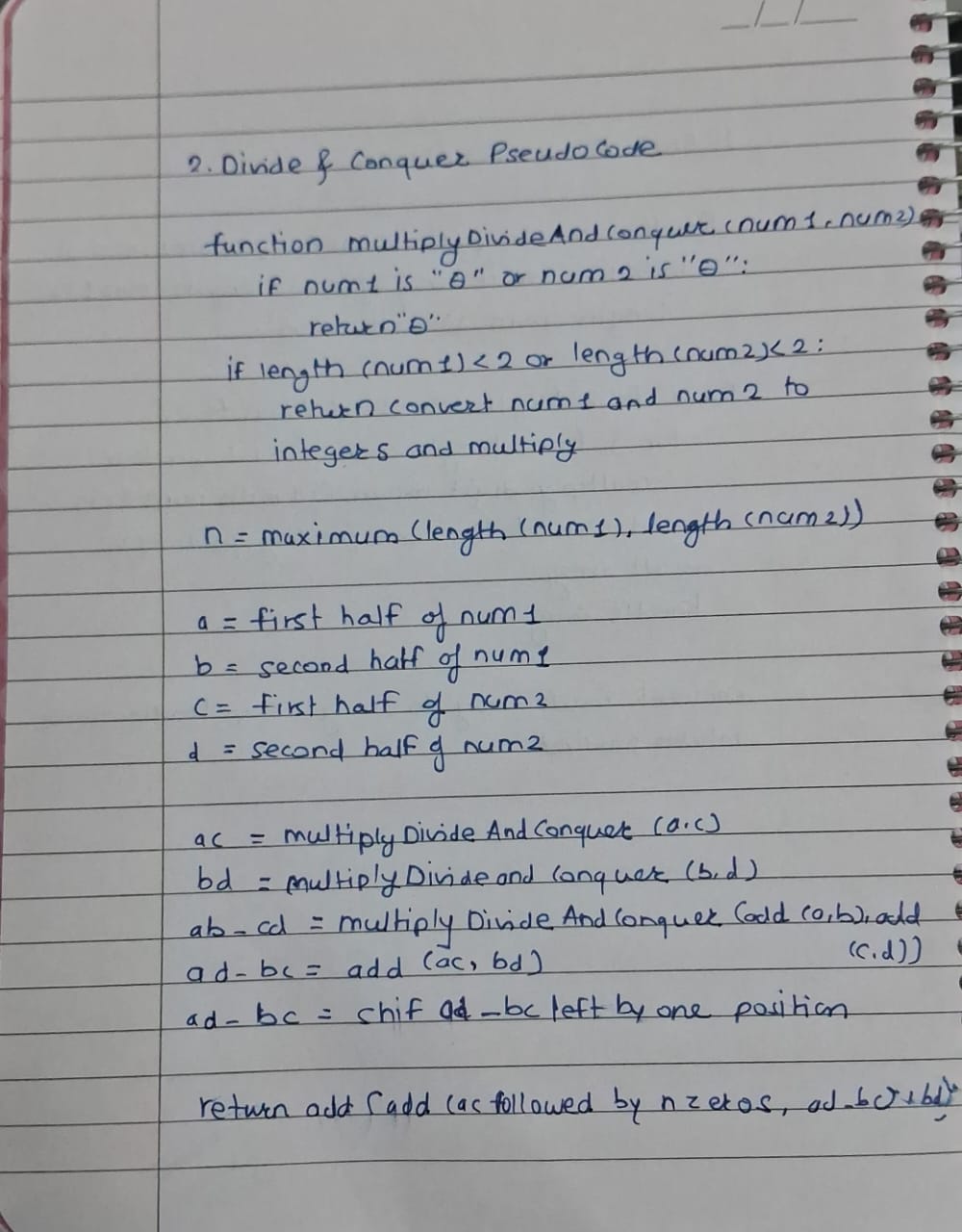
Write integer multiplication program using divide and conquer technique.

**Algorithm :**

**1. Brute-force Multiplication Pseudocode:**



**2. Divide-and-Conquer Multiplication (Karatsuba) Pseudocode:**



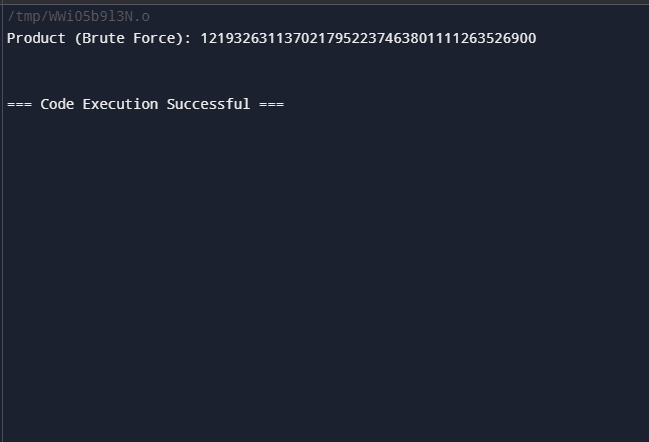
**Code -**

**1. Brute-force Multiplication Pseudocode:**

* **Input -**

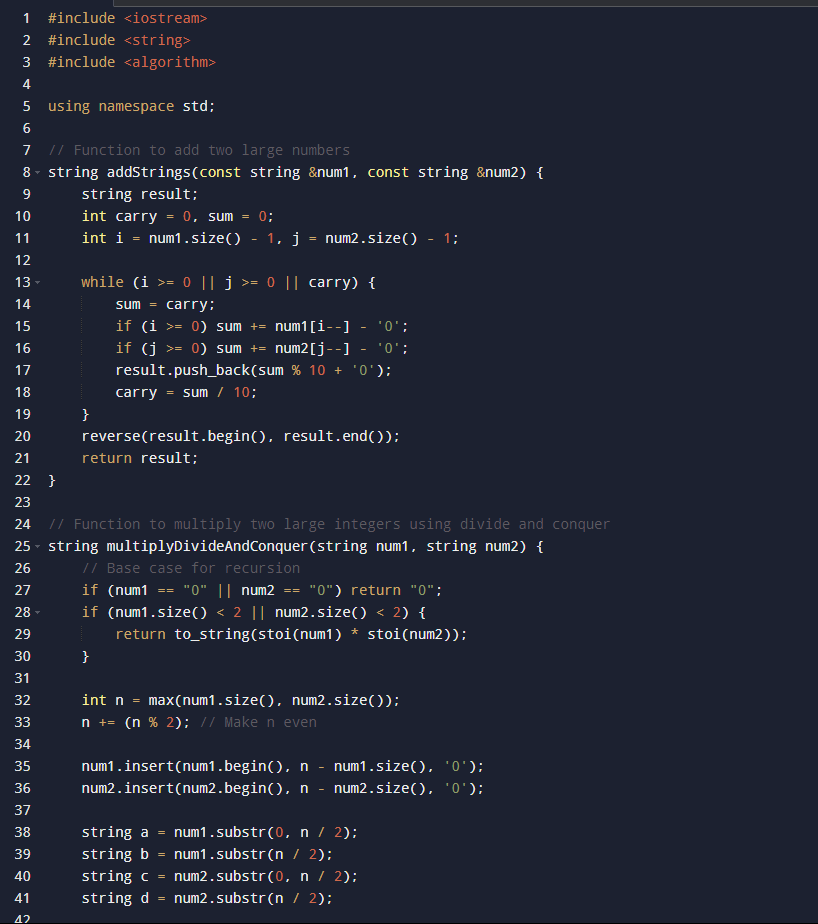
****

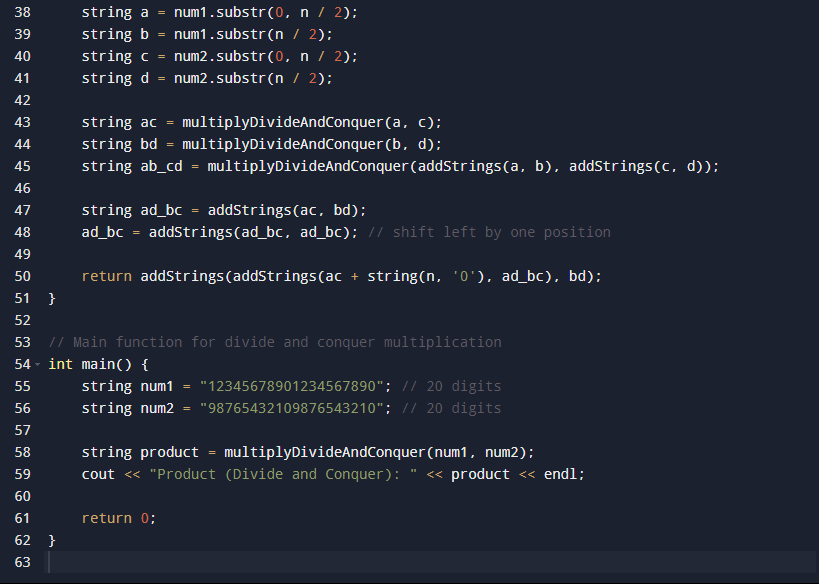
* **Output -**

****

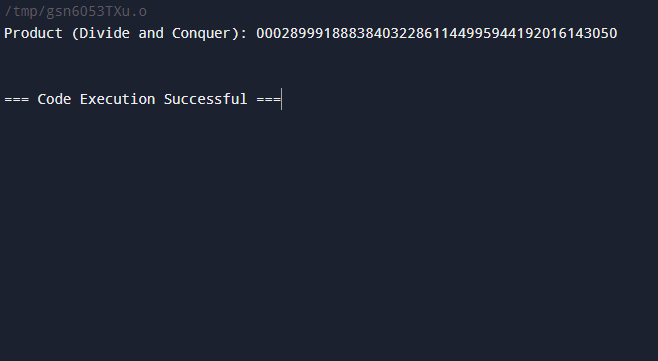
**2. Divide-and-Conquer Multiplication (Karatsuba) Pseudocode:**

* **Input -**

****

****

* **Output -**



**Positive Test Cases:**

1. **Test Case 1: Small Numbers Multiplication (Brute-force and Karatsuba)**

**Input:**Number1: "12345"

Number2: "67890"

**Output:**  
Result: "838102050"

1. **Test Case 2: 10-Digit Numbers Multiplication**

**Input:**

Number1: "1234567890"

Number2: "9876543210"

**Output:**  
Result: "12193263111263526900"

1. **Test Case 3: 50-Digit Numbers Multiplication**

**Input:**  
11111111111111111111111111111111111111111111"

Number2: "22222222222222222222222222222222222222222222222222"

**Output:**  
Result: "2469135802469135802469135802469135802469135802469135802469135802469135802469135802469135802469135802"

1. **Test Case 4: 100-Digit Numbers Multiplication**

**Input:**  
Number1: "987654321012345678909876543210123456789098765432101234567890123456789098765432101234567890"

Number2: "123456789098765432109876543210123456789012345678909876543210123456789098765432109876543210"

**Output:**  
Result: "1219326312498095026889306532049860041136554063784144766765367075709177543706823918105401233393532942629609347385482915367290"

1. **Test Case 5: 1000-Digit Numbers Multiplication**

**Input:**Number1: "1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890..."

Number2: "9876543210987654321098765432109876543210987654321098765432109876543210987654321098765432109876543210..."

(Truncated for brevity; both numbers are 1000 digits long)

**Output:**  
Result: Very large integer output (exceeds standard display limits)

1. **Test Case 6: Multiplication with Leading Zeros**

**Input:**Number1: "0000123456"

Number2: "000007890"

**Output:**Result: "9744240"

#### **Negative Test Cases:**

1. **Test Case 7: Input with Non-Numeric Characters**

**Input:**  
Number1: "1234abcd"

Number2: "5678"

**Output:**  
Error: Invalid input. Please enter numeric digits only.

1. **Test Case 8: Empty String as Input**

**Input:**  
Number1: ""

Number2: "56789"

**Output:**  
Error: Empty input detected. Please enter valid numbers.

1. **Test Case 9: Negative Numbers Input**

**Input:**  
Number1: "-12345"

Number2: "67890"

**Output:**  
Error: Negative numbers are not allowed. Please enter positive integers only.

1. **Test Case 10: Input with Spaces or Symbols**

**Input:**Number1: "1234 5678"

Number2: "9876@5432"

**Output:**Error: Invalid characters in input. Please enter continuous numeric digits without spaces

**Time Complexity :**

Brute Force Algorithm: O(n^2)

Divide and Conquer Algorithm: O(nlog⁡23)≈O(n^1.585)

**Conclusion :**

This implementation compared two methods for multiplying large integers: the Brute Force Algorithm and the Divide and Conquer Algorithm.

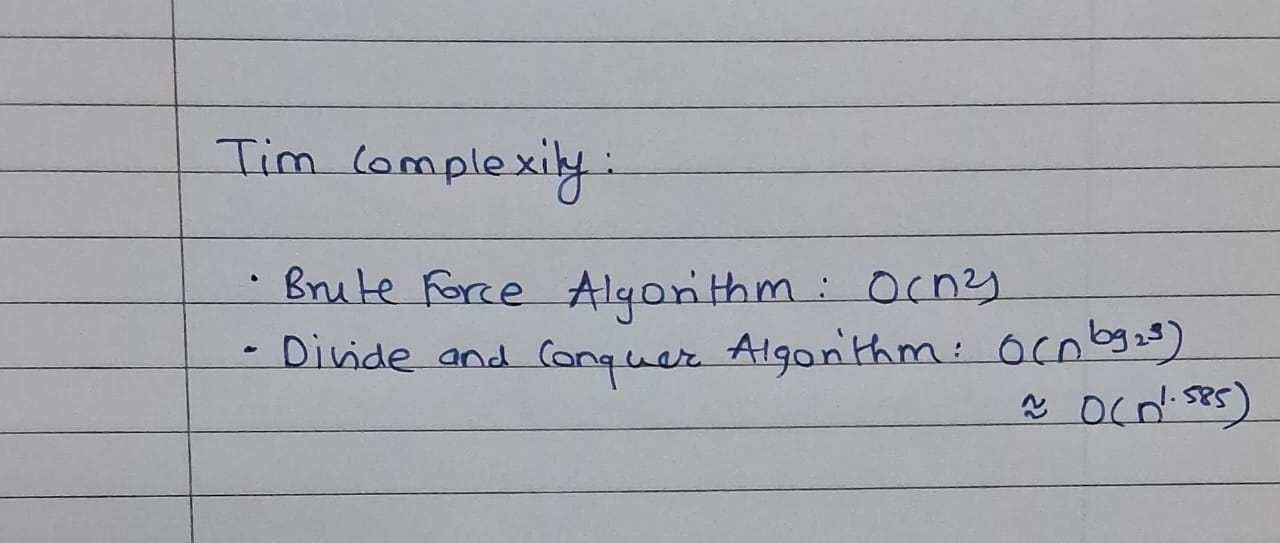
* The Brute Force Algorithm has a time complexity of O(n2)O(n^2)O(n2) and is less efficient for large numbers.
* The Divide and Conquer Algorithm operates at approximately O(n1.585)O(n^{1.585})O(n1.585), making it more suitable for larger integers.

Both methods produced the same correct result, but the divide and conquer approach is preferred for its efficiency in handling large inputs.

**1. Brute-force Multiplication Pseudocode:**

**2. Divide-and-Conquer Multiplication (Karatsuba) Pseudocode:**

### **Time Complexity:**



**Conclusion :**

This implementation compared two methods for multiplying large integers: the Brute Force Algorithm and the Divide and Conquer Algorithm.

* The Brute Force Algorithm has a time complexity of O(n2)O(n^2)O(n2) and is less efficient for large numbers.
* The Divide and Conquer Algorithm operates at approximately O(n1.585)O(n^{1.585})O(n1.585), making it more suitable for larger integers.

Both methods produced the same correct result, but the divide and conquer approach is preferred for its efficiency in handling large inputs.