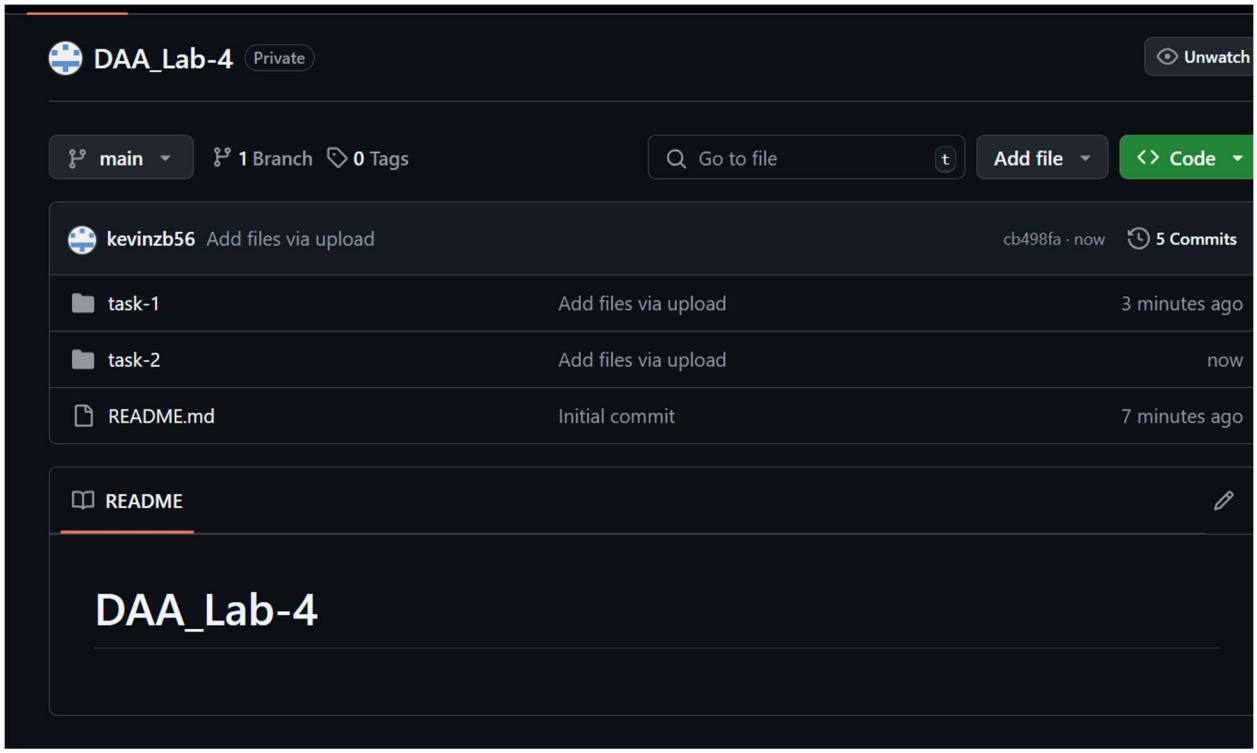
Kevin Shah
231070060 Batch-C

DAA ASSIGNMENT-4

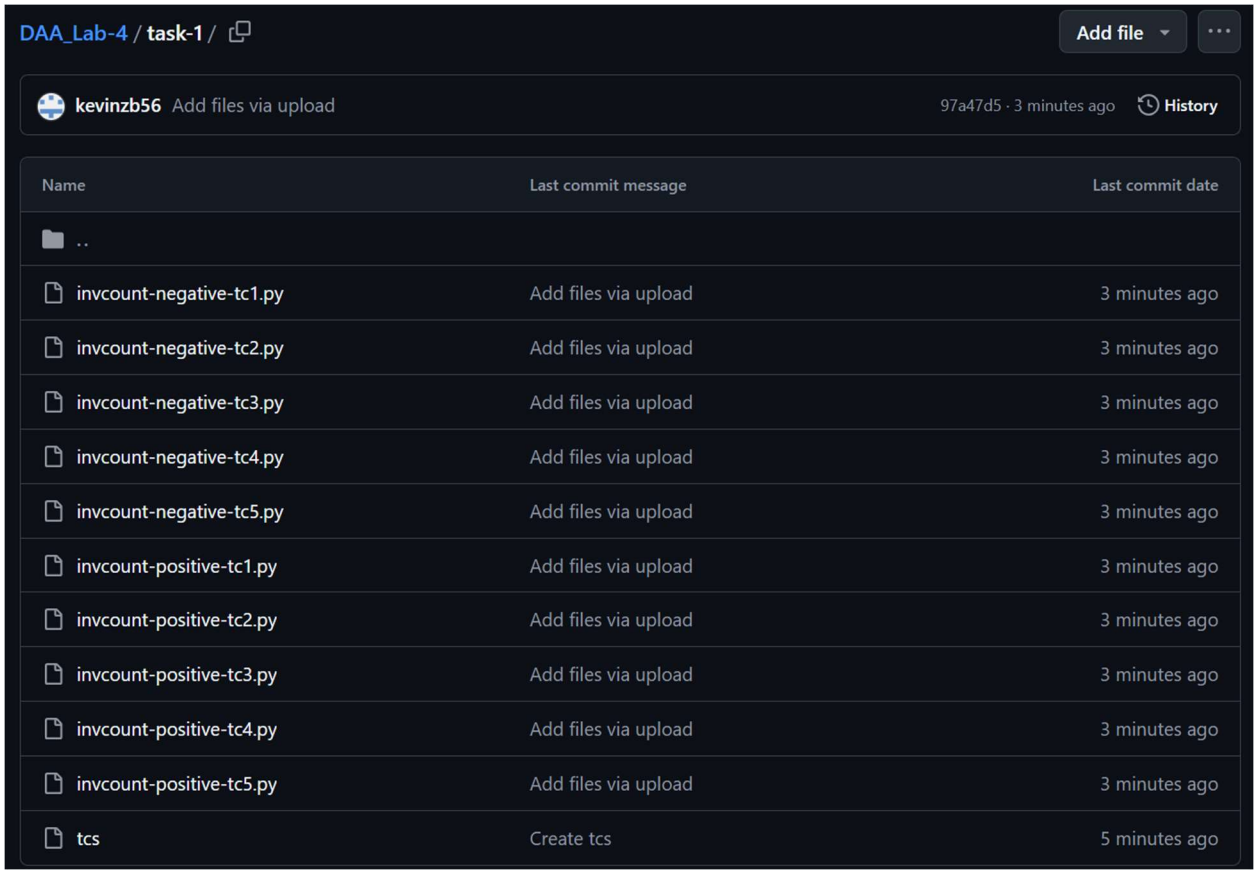
Task-1:

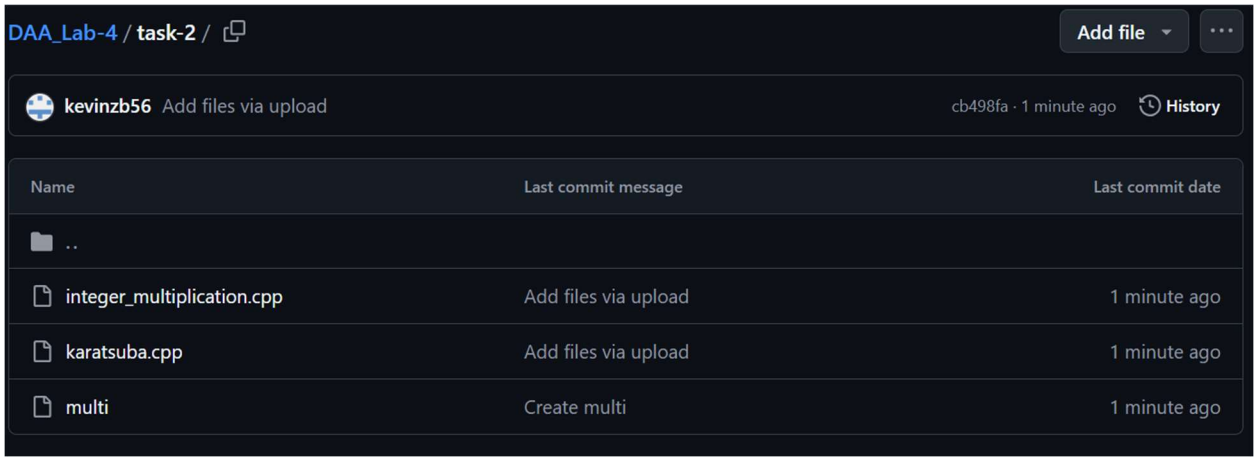
Aim:

The aim of this task is to understand the basics of source control systems by creating an account on GitHub, setting up a repository with a README file, learning the process of starting a new project, and pushing local files to the GitHub repository.



Kevin Shah
231070060 Batch-C





Link to the repo :- https://github.com/kevinzb56/DAA_Lab-4

Conclusion:

In this task, we successfully learned the fundamentals of using a source control system by creating a GitHub account, setting up a repository, and adding a README file. We also explored the process of initiating a new project and gained practical

experience in pushing local files to the GitHub repository, reinforcing the importance of version control in software development.

Experiment task-1:

Aim:

The aim of this project is to analyze the course-code choices of 100 first- and second-year students by calculating the inversion count for each student's selections. The experiment will classify students based on their inversion counts (zero, one, two, three or more) and provide insights on the patterns observed in their preferences.

Algorithm:

Kevin Shah
231070060 Batch-C

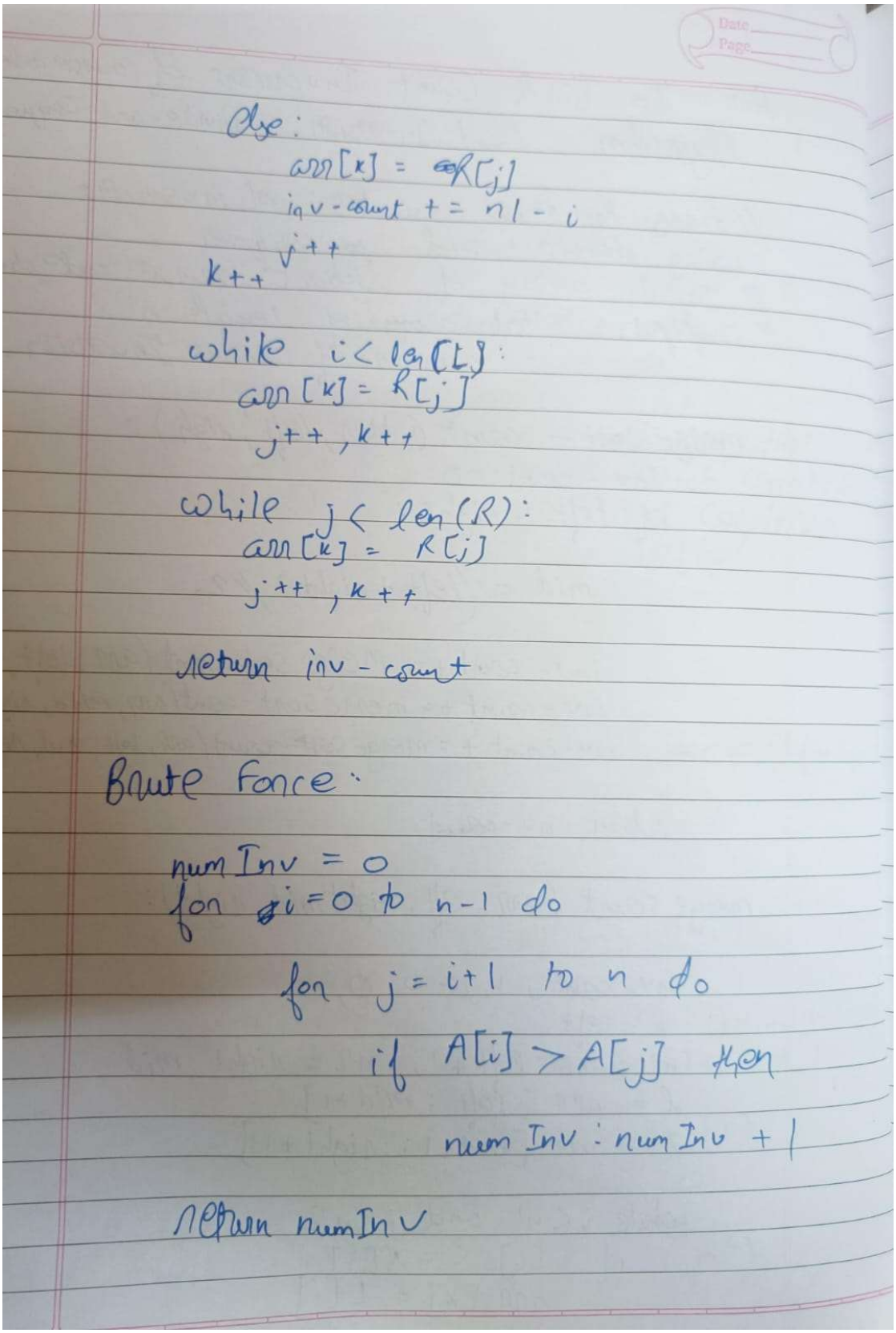
Aim: To find Count Inversions of course codes

1) Algorithm: Count Inversion (Divide-and-Conquer)

// finds the total number of inversions using divide and conquer
// Input: array of distinct course code choices
// Output: Sorted array of length n & number of count inversion.

```
merge sort - count ( arr, left, right ):
    inv - count = 0
    if left < right :

        mid = (left + right) // 2

        inv - count += merge sort - count (arr, left, mid)
        inv - count += merge sort - count (arr, mid+1, right)
        inv - count += merge ~~sort~~ - count (arr, left, mid, right)

    return inv - count

merge count (arr, left, ~~right~~ mid, right):

    inv - count , i, j = 0, 0, 0
    k = left
    n1 = mid - left + 1, n2 = right - mid
    L = arr [ left : mid + 1]
    R = arr [mid + 1 : right + 1]

    while i < n1 and j < n2
        if L[i] <= R[j]:
            arr [k] = L[i]
            i + +
```

Kevin Shah
231070060 Batch-C



```
else:
    arr[k] = R[j]
    inv-count += n1 - i
    j++
k++

while i < len(L):
    arr[k] = R[j]
    j++, k++

while j < len(R):
    arr[k] = R[j]
    j++, k++

return inv-count
```

Brute Force:

```
numInv = 0
for i = 0 to n-1 do
    for j = i+1 to n do
        if A[i] > A[j] then
            numInv : numInv + 1
return numInv
```

Kevin Shah
231070060 Batch-C

Test cases:

Positive:

1]

```
:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn
 4\task1\invcount-positive-tc1.py"
Total inversion count (Brute Force) across all students: 244
Total inversion count (Divide and Conquer) across all students: 244

Categorized Inversion Counts (Brute Force):
Inversion Count 0: Students [12, 18, 27, 29, 38, 41, 45, 49, 55, 61, 90, 94, 97, 98, 99]
Inversion Count 1: Students [2, 5, 13, 21, 22, 33, 37, 46, 58, 78, 85, 87, 88, 95]
Inversion Count 3: Students [6, 8, 11, 14, 34, 36, 39, 43, 44, 48, 57, 62, 63, 64, 65, 70, 71, 72, 76, 80, 93]
Inversion Count 4: Students [7, 9, 10, 17, 24, 30, 32, 47, 51, 59, 67, 74, 75, 89, 91, 96]
Inversion Count 5: Students [16, 20, 26, 28, 31, 52, 53, 68, 82]
Inversion Count 6: Students [3, 42]

Categorized Inversion Counts (Divide and Conquer):
Inversion Count 0: Students [12, 18, 27, 29, 38, 41, 45, 49, 55, 61, 90, 94, 97, 98, 99]
Inversion Count 1: Students [2, 5, 13, 21, 22, 33, 37, 46, 58, 78, 85, 87, 88, 95]
Inversion Count 2: Students [1, 4, 15, 19, 23, 25, 35, 40, 50, 54, 56, 60, 66, 69, 73, 77, 79, 81, 83, 84, 86, 9
2, 100]
Inversion Count 3: Students [6, 8, 11, 14, 34, 36, 39, 43, 44, 48, 57, 62, 63, 64, 65, 70, 71, 72, 76, 80, 93]
Inversion Count 4: Students [7, 9, 10, 17, 24, 30, 32, 47, 51, 59, 67, 74, 75, 89, 91, 96]
Inversion Count 5: Students [16, 20, 26, 28, 31, 52, 53, 68, 82]
Inversion Count 6: Students [3, 42]
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

2]

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\A
ssgn 4\task1\invcount-positive-tc2.py"
Total inversion count (Brute Force) across all students: 260
Total inversion count (Divide and Conquer) across all students: 260
Inversion Count 1: Students [2, 8, 19, 27, 30, 38, 40, 41, 45, 48, 61, 98, 99]
Inversion Count 2: Students [3, 5, 10, 14, 15, 16, 21, 23, 28, 34, 39, 46, 50, 51, 59, 76, 78, 81, 86, 87, 89, 9
2, 93, 95]
Inversion Count 3: Students [4, 7, 9, 11, 17, 20, 24, 26, 37, 42, 44, 47, 49, 58, 63, 66, 67, 68, 69, 70, 71, 73
, 74, 75, 77, 79, 94]
Inversion Count 4: Students [12, 25, 29, 31, 35, 43, 53, 55, 56, 60, 65, 82, 84, 88, 90, 91, 96, 97, 100]
Inversion Count 5: Students [22, 32, 57, 62, 64, 72]
Inversion Count 6: Students [18, 85]

Categorized Inversion Counts (Divide and Conquer):
Inversion Count 0: Students [1, 6, 13, 33, 36, 52, 54, 80, 83]
Inversion Count 1: Students [2, 8, 19, 27, 30, 38, 40, 41, 45, 48, 61, 98, 99]
Inversion Count 2: Students [3, 5, 10, 14, 15, 16, 21, 23, 28, 34, 39, 46, 50, 51, 59, 76, 78, 81, 86, 87, 89, 9
2, 93, 95]
Inversion Count 3: Students [4, 7, 9, 11, 17, 20, 24, 26, 37, 42, 44, 47, 49, 58, 63, 66, 67, 68, 69, 70, 71, 73
, 74, 75, 77, 79, 94]
Inversion Count 4: Students [12, 25, 29, 31, 35, 43, 53, 55, 56, 60, 65, 82, 84, 88, 90, 91, 96, 97, 100]
Inversion Count 5: Students [22, 32, 57, 62, 64, 72]
Inversion Count 6: Students [18, 85]
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

3]

Kevin Shah
231070060 Batch-C

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-positive-tc3.py"
Total inversion count (Brute Force) across all students: 256
Total inversion count (Divide and Conquer) across all students: 256

Categorized Inversion Counts (Brute Force):
Inversion Count 0: Students [23, 53, 57, 88, 95]
Inversion Count 1: Students [10, 12, 18, 22, 30, 34, 36, 44, 46, 58, 62, 74, 83, 86, 92, 99, 100]
Inversion Count 2: Students [1, 3, 4, 5, 25, 27, 28, 31, 33, 39, 41, 50, 51, 59, 68, 72, 73, 79, 81, 85, 89, 90, 97]
Inversion Count 3: Students [6, 7, 8, 11, 13, 14, 17, 20, 24, 26, 29, 32, 37, 38, 45, 47, 48, 49, 52, 55, 60, 61, 66, 69, 71, 75, 76, 77, 80, 82, 87, 91, 93, 94, 98]
Inversion Count 4: Students [2, 9, 15, 16, 19, 21, 42, 54, 65, 67, 70, 78, 84]
Inversion Count 5: Students [35, 43, 56, 63, 64, 96]
Inversion Count 6: Students [40]

Categorized Inversion Counts (Divide and Conquer):
Inversion Count 0: Students [23, 53, 57, 88, 95]
Inversion Count 1: Students [10, 12, 18, 22, 30, 34, 36, 44, 46, 58, 62, 74, 83, 86, 92, 99, 100]
Inversion Count 2: Students [1, 3, 4, 5, 25, 27, 28, 31, 33, 39, 41, 50, 51, 59, 68, 72, 73, 79, 81, 85, 89, 90, 97]
Inversion Count 3: Students [6, 7, 8, 11, 13, 14, 17, 20, 24, 26, 29, 32, 37, 38, 45, 47, 48, 49, 52, 55, 60, 61, 66, 69, 71, 75, 76, 77, 80, 82, 87, 91, 93, 94, 98]
Inversion Count 4: Students [2, 9, 15, 16, 19, 21, 42, 54, 65, 67, 70, 78, 84]
Inversion Count 5: Students [35, 43, 56, 63, 64, 96]
Inversion Count 6: Students [40]
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

4]

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-positive-tc4.py"
Total inversion count (Brute Force) across all students: 294
Total inversion count (Divide and Conquer) across all students: 294

Categorized Inversion Counts (Brute Force):
Inversion Count 0: Students [6, 16, 30, 67, 100]
Inversion Count 1: Students [7, 13, 17, 39, 53, 54, 57]
Inversion Count 2: Students [9, 10, 14, 15, 22, 27, 31, 33, 34, 41, 43, 48, 49, 50, 51, 52, 60, 61, 73, 78, 86, 87, 89, 92, 93, 96, 98]
Inversion Count 3: Students [1, 3, 11, 12, 23, 24, 25, 26, 29, 36, 38, 40, 55, 59, 63, 72, 74, 75, 76, 83, 85, 88, 90, 91, 94, 99]
Inversion Count 4: Students [2, 4, 18, 20, 21, 32, 35, 44, 45, 46, 56, 58, 65, 66, 68, 70, 71, 80, 81, 82, 84, 95]
Inversion Count 5: Students [5, 8, 19, 28, 37, 42, 47, 62, 64, 79, 97]
Inversion Count 6: Students [69, 77]

Categorized Inversion Counts (Divide and Conquer):
Inversion Count 0: Students [6, 16, 30, 67, 100]
Inversion Count 1: Students [7, 13, 17, 39, 53, 54, 57]
Inversion Count 2: Students [9, 10, 14, 15, 22, 27, 31, 33, 34, 41, 43, 48, 49, 50, 51, 52, 60, 61, 73, 78, 86, 87, 89, 92, 93, 96, 98]
Inversion Count 3: Students [1, 3, 11, 12, 23, 24, 25, 26, 29, 36, 38, 40, 55, 59, 63, 72, 74, 75, 76, 83, 85, 88, 90, 91, 94, 99]
Inversion Count 4: Students [2, 4, 18, 20, 21, 32, 35, 44, 45, 46, 56, 58, 65, 66, 68, 70, 71, 80, 81, 82, 84, 95]
Inversion Count 5: Students [5, 8, 19, 28, 37, 42, 47, 62, 64, 79, 97]
Inversion Count 6: Students [69, 77]
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

5]

Kevin Shah
231070060 Batch-C

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code

PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-positive-tc5.py"
Total inversion count (Brute Force) across all students: 272
Total inversion count (Divide and Conquer) across all students: 272

Categorized Inversion Counts (Brute Force):
Inversion Count 0: Students [1, 9, 21, 27, 39, 57, 80, 88]
Inversion Count 1: Students [3, 14, 41, 46, 48, 78, 82, 83, 85, 92, 95]
Inversion Count 2: Students [2, 6, 7, 17, 19, 20, 22, 25, 29, 33, 35, 36, 40, 47, 50, 60, 61, 67, 71, 75, 79, 84, 100]
Inversion Count 3: Students [4, 5, 8, 12, 13, 16, 18, 28, 31, 34, 43, 45, 49, 51, 55, 63, 65, 66, 68, 69, 72, 74, 76, 89, 91, 93, 97, 99]
Inversion Count 4: Students [10, 15, 23, 24, 30, 32, 37, 42, 44, 52, 53, 56, 58, 62, 73, 77, 81, 86, 98]
Inversion Count 5: Students [11, 26, 38, 54, 59, 64, 70, 87, 90, 94, 96]

Categorized Inversion Counts (Divide and Conquer):
Inversion Count 0: Students [1, 9, 21, 27, 39, 57, 80, 88]
Inversion Count 1: Students [3, 14, 41, 46, 48, 78, 82, 83, 85, 92, 95]
Inversion Count 2: Students [2, 6, 7, 17, 19, 20, 22, 25, 29, 33, 35, 36, 40, 47, 50, 60, 61, 67, 71, 75, 79, 84, 100]
Inversion Count 3: Students [4, 5, 8, 12, 13, 16, 18, 28, 31, 34, 43, 45, 49, 51, 55, 63, 65, 66, 68, 69, 72, 74, 76, 89, 91, 93, 97, 99]
Inversion Count 4: Students [10, 15, 23, 24, 30, 32, 37, 42, 44, 52, 53, 56, 58, 62, 73, 77, 81, 86, 98]
Inversion Count 5: Students [11, 26, 38, 54, 59, 64, 70, 87, 90, 94, 96]
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

Negative:

1]

```
63  # The provided nested list
64  students_random_numbers = []
65

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code

ERROR: The list of course code is empty, so the inversion count cannot be found by neither brute force nor divide and conquer approach.
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

2]

```
70  # The provided nested list (contains a mix of integers and letters for testing)
71  students_random_numbers = [
72      [5, 2, 3, 6], ['a', 1, 5, 2], [7, 6, 4, 1], [6, 2, 'b', 7], [2, 3, 8, 4], [5, 5, 5, 4]
73  ]
74

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code

PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-negative-tc2.py"

Categorized Inversion Counts (Valid Entries):
Student 1: Brute Force Inversion Count = 2, Divide and Conquer Inversion Count = 2
Student 3: Brute Force Inversion Count = 6, Divide and Conquer Inversion Count = 6
Student 5: Brute Force Inversion Count = 1, Divide and Conquer Inversion Count = 1
Student 6: Brute Force Inversion Count = 3, Divide and Conquer Inversion Count = 3

Error Messages for Invalid Entries:
Student 2: Error: Array contains non-integer values, inversion count can't be performed.
Student 4: Error: Array contains non-integer values, inversion count can't be performed.
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

Kevin Shah
231070060 Batch-C

3]

```
70  # The provided nested list (contains a mix of integers and letters for testing)
71  students_random_numbers = [
72      [5, 2, 3, 6], ['a', 1, 5, 2], [-7, -6, -4, -1], [-6, -2, -5, -7], [2, 3, 8, 4], [5, 5, 5
73  ]
74
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-negative-tc3.py"

Categorized Inversion Counts (Valid Entries):
Student 1: Brute Force Inversion Count = 2, Divide and Conquer Inversion Count = 2
Student 5: Brute Force Inversion Count = 1, Divide and Conquer Inversion Count = 1
Student 6: Brute Force Inversion Count = 3, Divide and Conquer Inversion Count = 3

Negative Integer Entries:
Student 3: Inversion count can be found since course code cant be negative.
Student 4: Inversion count can be found since course code cant be negative.

Error Messages for Invalid Entries:
Student 2: Error: Array contains letters instead of integer values, inversion count can't be performed.
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

4]

```
70  # The provided nested list (contains a mix of integers and letters for testing)
71  students_random_numbers = [
72      [5, 2, 3, 6], [-3, -1, -5, -2], [-7, -6, -4, -1], [-6, -2, -5, -7], [2, 3, 8, 4], [5, 5,
73  ]
74
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-negative-tc4.py"

Categorized Inversion Counts (Valid Entries):
Student 1: Brute Force Inversion Count = 2, Divide and Conquer Inversion Count = 2
Student 5: Brute Force Inversion Count = 1, Divide and Conquer Inversion Count = 1
Student 6: Brute Force Inversion Count = 3, Divide and Conquer Inversion Count = 3

Negative Integer Entries:
Student 2: Inversion count can be found since course code cant be negative.
Student 3: Inversion count can be found since course code cant be negative.
Student 4: Inversion count can be found since course code cant be negative.
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

5]

Kevin Shah
231070060 Batch-C

```
70  # The provided nested list (contains a mix of integers, letters, and booleans for testing)
71  students_random_numbers = [
72      [5, 2, 3, 6], [-2, -1, -5, -2], [7, False, 4, 1], [6, 2, -5, -7],
73      [2, 3, 8, 4], [5, 5, 5, 4], [True, False, 1, 0]
74  ]
75
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> python -u "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\task1\invcount-negative-tc5.py"

Categorized Inversion Counts (Valid Entries):
Student 1: Brute Force Inversion Count = 2, Divide and Conquer Inversion Count = 2
Student 4: Brute Force Inversion Count = 6, Divide and Conquer Inversion Count = 6
Student 5: Brute Force Inversion Count = 1, Divide and Conquer Inversion Count = 1
Student 6: Brute Force Inversion Count = 3, Divide and Conquer Inversion Count = 3

Negative Integer Entries:
Student 2: ERROR: Inversion count can't be found since course code can't be negative.

Error Messages for Invalid Entries:
Student 3: ERROR: Inversion count can't be found due to the presence of boolean values.
Student 7: ERROR: Inversion count can't be found due to the presence of boolean values.
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

Handwritten test cases and Time Complexity:

Kevin Shah
231070060 Batch-C

Example: Base = [1, 2, 3, 4, 5]

1. [5, 4, 3, 2, 1] → count = 10
2. [4, 3, 2, 1, 5] → count = 6
3. [1, 2, 3, 5, 4] → count = 1
4. [1, 2, 3, 4, 5] → count = 0
5. [5, 3, 4, 1, 2] → count = 8
6. [1, 2, 3, 4] → Not compatible
7. [1, 2, 3, 'a', 5] → Not compatible
8. [-1, 2, 3, 4, 5] → Not compatible
9. [True, 2, 3, 4, False] → Not compatible
10. [] → Not compatible

Time Complexity :

Divide-And-Conquer

$T(n) = 2T(n/2) + n$

- 2 subproblems of size $n/2$
- merge of sorted array and counting

$\therefore T(n) = aT(n/b) + f(n)$

$\therefore a = 2, b = 2, d = 1$

$\because b^d = a \quad \{\because 2^1 = 2\}$

⇒ Time complexity : $n^d \log n$

$O(n \log n)$

Brute Force (2 for loops)

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n} 1$$

$$= \sum_{i=0}^{n-1} (n - i)$$

$$= (n-0) + (n-1) + \ldots (n-(n-1))$$

$$= n + (n-1) + \ldots + 1$$

$$= \frac{n(n+1)}{2}$$

⇒ $O(n^2)$

Kevin Shah
231070060 Batch-C

Program

1)Brute Force

```
# Brute force method to count the number of inversions in a list
def count_inversions_brute_force(arr):
    inversions = 0
    n = len(arr)

    # Compare each pair (i, j) where i < j
    for i in range(n):
        for j in range(i + 1, n):
            if arr[i] > arr[j]:
                inversions += 1

    return inversions

# Example usage
arr = [8, 4, 2, 1]

# Count inversions using brute force
brute_force_inversions = count_inversions_brute_force(arr)
print(f"Brute Force Inversions: {brute_force_inversions}")
```

2)Divide-and-Conquer

```
# Divide and conquer (merge sort) method to count the number of inversions
in a list
def count_inversions_divide_and_conquer(arr):
    return merge_sort(arr, 0, len(arr) - 1)

def merge_sort(arr, left, right):
    if left >= right:
        return 0

    mid = (left + right) // 2
    inversions = merge_sort(arr, left, mid)
    inversions += merge_sort(arr, mid + 1, right)
    inversions += merge_and_count(arr, left, mid, right)
```

Kevin Shah
231070060 Batch-C

```
    return inversions

def merge_and_count(arr, left, mid, right):
    # Left and right subarrays
    left_subarray = arr[left:mid + 1]
    right_subarray = arr[mid + 1:right + 1]

    i = j = 0
    k = left
    inversions = 0

    # Merge while counting inversions
    while i < len(left_subarray) and j < len(right_subarray):
        if left_subarray[i] <= right_subarray[j]:
            arr[k] = left_subarray[i]
            i += 1
        else:
            arr[k] = right_subarray[j]
            inversions += (mid - i + 1 - left)  # Count inversions
            j += 1
        k += 1

    # Copy remaining elements
    while i < len(left_subarray):
        arr[k] = left_subarray[i]
        i += 1
        k += 1

    while j < len(right_subarray):
        arr[k] = right_subarray[j]
        j += 1
        k += 1

    return inversions


# Example usage
arr = [8, 4, 2, 1]

# Count inversions using divide and conquer
```

```
divide_and_conquer_inversions = count_inversions_divide_and_conquer(arr)
print(f"Divide and Conquer Inversions: {divide_and_conquer_inversions}")
```

Experiment task-2:

Algorithm and Test Cases:

Kevin Shah
231070060 Batch-C

Multiplication of 2 integers

Aim: To multiple 2 integers given by the user and output it.

Algorithm: ~~Brut~~ (Brute Force)

// Input : 2 integers
// Output : Multiplicative product

Mul (a, b) :

```
a ← min(a, b)
y ← max(a, b)
ans ← 0
i ← 0
for each bit of y do
    partial ← bit x x
    partial ← partial x 10^i
    i ← i + 1

    ans ← ans + partial

return ans
```

Algorithm : Divide-and-conquer (Karatsuba)

// Input : 2 integers
// Output : Multiplication answer of 2 integers

Kevin Shah
231070060 Batch-C

mul(a, b):

if a < 10 or b < 10:
    return a×b

smaller multiplican can be done easily

n1 ← no. of digits in a
n2 ← no. of digits in b

n ← max(n1, n2) // 2

x1 ← a // 10^n
x0 ← a % 10^n
y1 ← b // 10^n
y0 ← b % 10^n

p = mul(x1, y1)
q = mul(x0, y0)
r = mul(x1 + x0, y1 + y0) − p − q

return $p \times 10^{2n} + r \times 10^n + q$

Test cases :

1) Positive :

① a = 7891105
   b = 4554210 → 35937749302050

Kevin Shah
231070060 Batch-C

(2) a = 12345 → 838102050
b = 67890

(3) a = 11223344 → 624,778,734,443,072
b = 55667788

(4) a = 857611 → 676827,458,811
b = 789201

(5) a = 705222 → 691,925,624,87,288
b = 982104

(6) a = 137.56 → Not valid
b = 223

(7) a = 1375.6 → Not valid
b = 145.88

(8) a = 1345 → Not valid
b = -

(9) a = -
b = - → Not valid

(10) a = 2.3 → Not valid
b = -1.8

Kevin Shah
231070060 Batch-C

Time Complexity:

Time complexity: (Brute force)

basic operation: Addition of 2 bits

Each integer is 'n' bit long

Therefore, the loop runs 'n' times and in the loop the line where the partial products are added also costs $O(n)$ time as there are n additions to be performed.

$\therefore$ Time complexity $= O(n^2)$

Time Complexity: (Karatsuba)

$T(n) = 3T(n/2) + n$

3 subproblems of length $n/2$; addition of f, g, h

$\therefore T(n) = aT(n/b) + f(n)$

$\therefore a = 3, \quad b = 2, \quad f(n) = n^1 = n^d \Rightarrow d = 1$

by master's theorem,

time complexity $= O(n^{\log_2 3})$

$\because b^d = 2^1 = 2 < a$

Kevin Shah
231070060 Batch-C

Outputs:

Integer Multiplication(school method):

```
48      // Input two large numbers as strings
49      string num1 = "12345678901234567890";  // 20-digit number
50      string num2 = "98765432109876543210";  // 20-digit number
51
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4> cd "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4\
" ; if ($?) { g++ integer_multiplication.cpp -o integer_multiplication } ; if ($?) { .\integer_multiplication }
Multiplication of 12345678901234567890 and 98765432109876543210:
1219326311370217952237463801111263526900
PS C:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA\Assgn 4>
```

Divide-and-Conquer(Karatsuba):

```
84  int main()
85  {
86      cout<<"\n";
87      cout << "Using karatsuba : " << "\n";
88      string num1_10_1 = "1234567890";
89      string num2_10_1 = "9876543210";
90      cout << "Test Case 1 (10 digits):" << endl;
91      cout << "Result: " << karatsuba(num1_10_1, num2_10_1) << endl;
92
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
                                                cd "c:\Users\Kevin Shah\OneDrive\Desktop\SY\DAA
" ; if ($?) { g++ karatsuba.cpp -o karatsuba } ; if ($?) { .\karatsuba }

Using karatsuba :
Test Case 1 (10 digits):
Result: 22457258295667746900
```

Kevin Shah
231070060 Batch-C

```
84  int main()
94      string num1_10_2 = "1029384756";
95      string num2_10_2 = "5647382910";
96      cout << "Test Case 2 (10 digits):" << endl;
97      cout << "Result: " << karatsuba(num1_10_2, num2_10_2) << endl;
98
99      cout<<"\n";
100     string num1_50_1 = "12345678901234567890123456789012345678901234567890";
101     string num2_50_1 = "98765432109876543210987654321098765432109876543210";
102     cout << "Test Case 3 (50 digits):" << endl;
103     cout << "Result: " << karatsuba(num1_50_1, num2_50_1) << endl;
104
105     cout<<"\n";
106     string num1_50_2 = "10293847561029384756102938475610293847561029384756";
107     string num2_50_2 = "56473829105647382910564738291056473829105647382910";
108     cout << "Test Case 4 (50 digits):" << endl;
109     cout << "Result: " << karatsuba(num1_50_2, num2_50_2) << endl;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

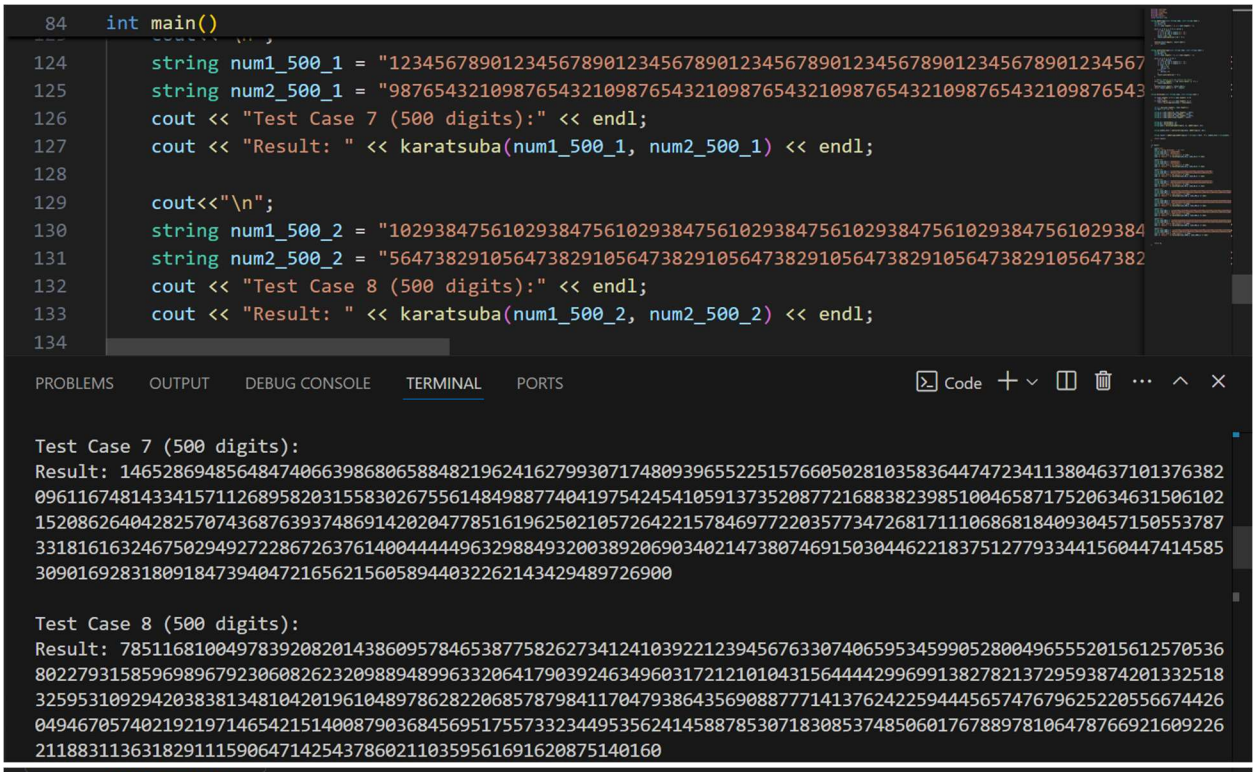
```
Test Case 2 (10 digits):
Result: 15853829539268940160

Test Case 3 (50 digits):
Result: 144518025849449732991439985732698524510454518647100039041352254166068378943505759464760558528972690 0

Test Case 4 (50 digits):
Result: 80350903966303039097029414433541148351989492478988830616519723002395077128239858332821187527514016 0
```

```
110
111     cout<<"\n";
112     string num1_100_1 = "1234567890123456789012345678901234567890123456789012345678901234567
113     string num2_100_1 = "9876543210987654321098765432109876543210987654321098765432109876543
114     cout << "Test Case 5 (100 digits):" << endl;
115     cout << "Result: " << karatsuba(num1_100_1, num2_100_1) << endl;
116
117     cout<<"\n";
118     string num1_100_2 = "1029384756102938475610293847561029384756102938475610293847561029384
119     string num2_100_2 = "5647382910564738291056473829105647382910564738291056473829105647382
120     cout << "Test Case 6 (100 digits):" << endl;
121     cout << "Result: " << karatsuba(num1_100_2, num2_100_2) << endl;
122
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

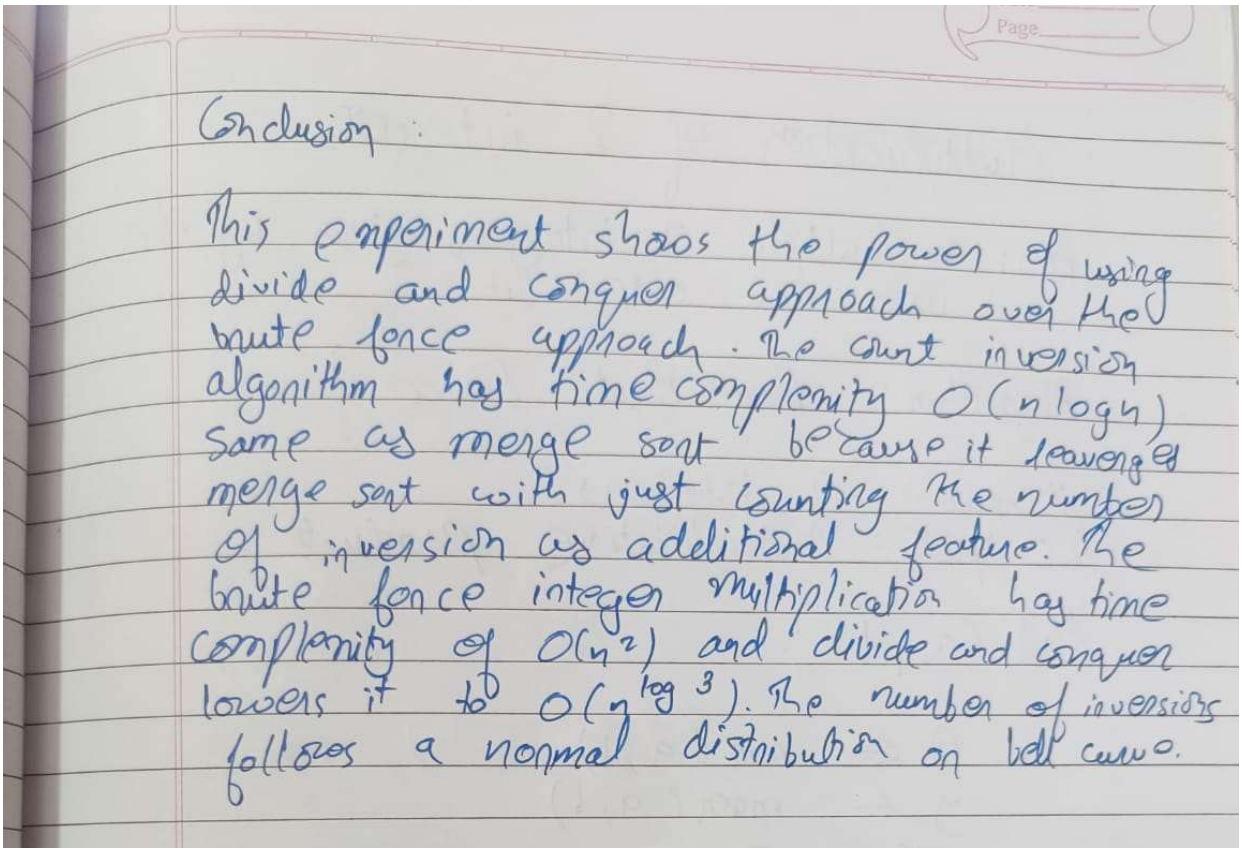
```
Test Case 5 (100 digits):
Result: 144518025849449732991439985732698524510454518647306875405490558128598157580438976219181397818916806810130336485403650415605378199833884415541210940510003904135225416606837894350575946476055852897269000

Test Case 6 (100 digits):
Result: 80350903966303039097029414433541148351989492478999171975434041146805107082206370932381250275179915527190336166734745629032131324953469544748383067418883061651972300239507712823985833282118752751401600
```

Kevin Shah
231070060 Batch-C



Conclusion:

Kevin Shah
231070060 Batch-C



Conclusion :

This experiment shows the power of using divide and conquer approach over the brute force approach. The count inversion algorithm has time complexity $O(n \log n)$ same as merge sort because it leverages merge sort with just counting the number of inversion as additional feature. The brute force integer multiplication has time complexity of $O(n^2)$ and divide and conquer lowers it to $O(n^{\log 3})$. The number of inversions follows a normal distribution on bell curve.