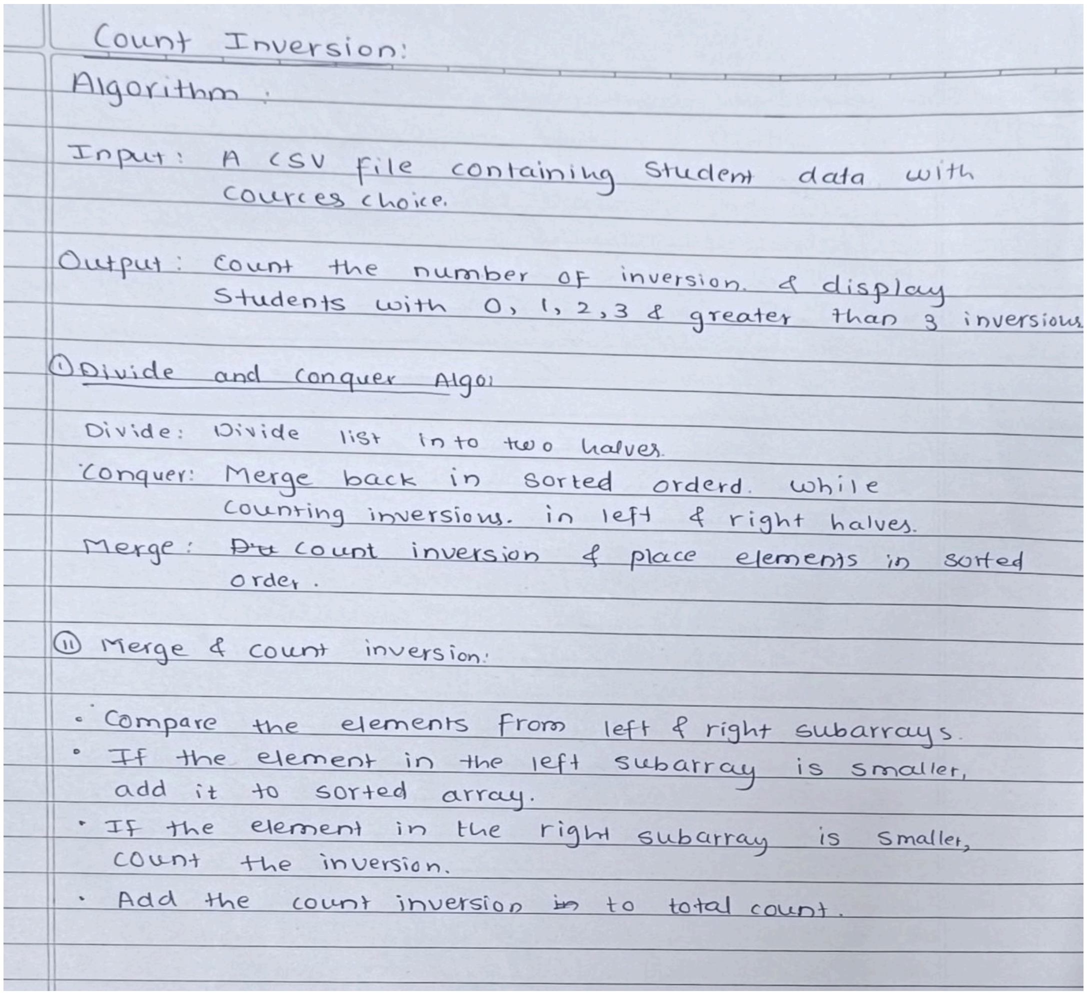
Name: Bhakti Anil Baraf
ID: 241071407
B.Tech SY computer engineering

Practical No. 4

Experiment task 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 and comment on your result.

Algorithm:



Count Inversion:

Algorithm:

Input: A CSV file containing student data with cources choice.

Output: Count the number of inversion. & display Students with 0, 1, 2, 3 & greater than 3 inversions

(i) Divide and conquer Algo:

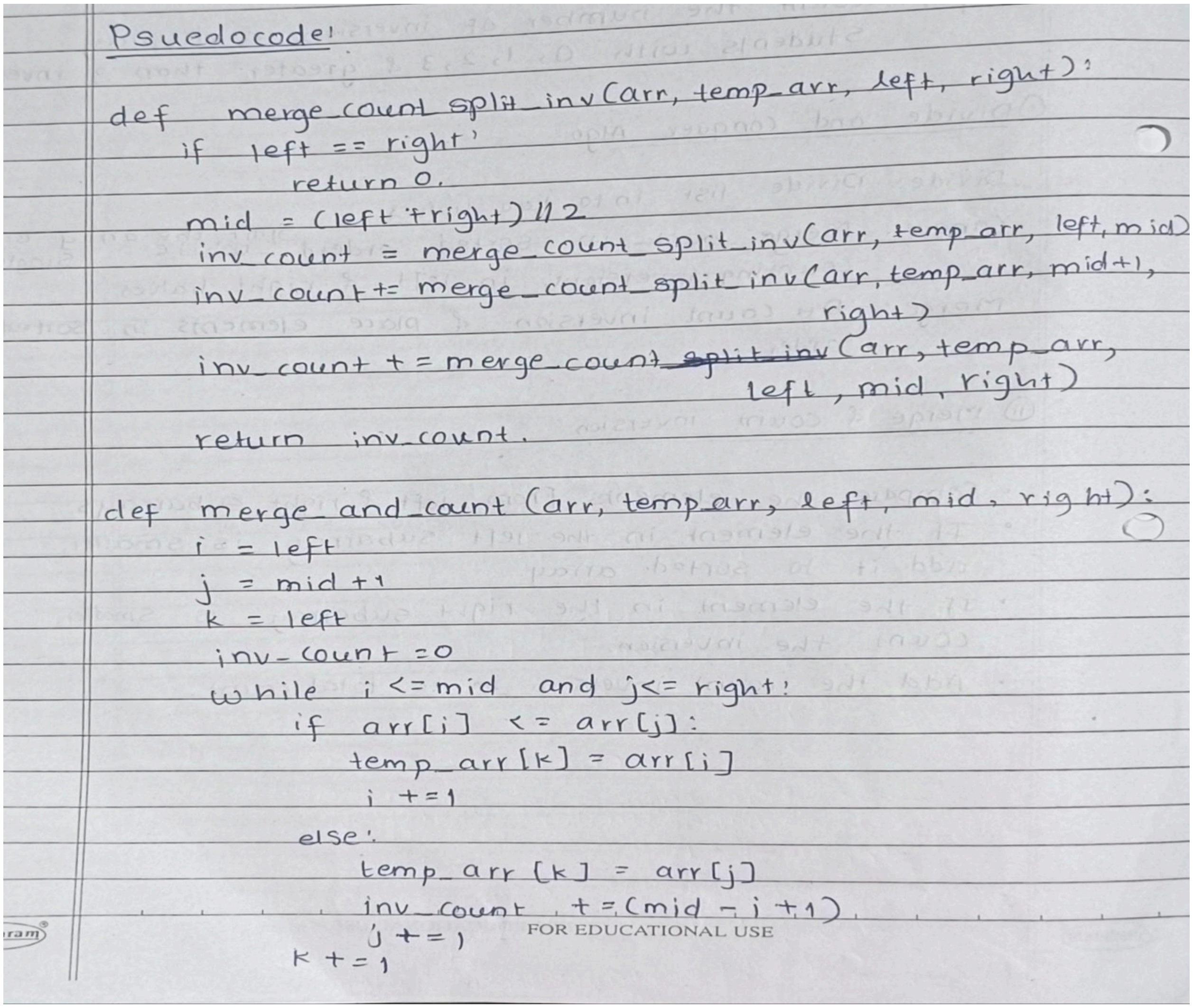
Divide: Divide list in to two halves.

Conquer: Merge back in sorted orderd. while counting inversions. in left & right halves.

Merge: ~~Dt~~ count inversion & place elements in sorted order.

(ii) Merge & count inversion:

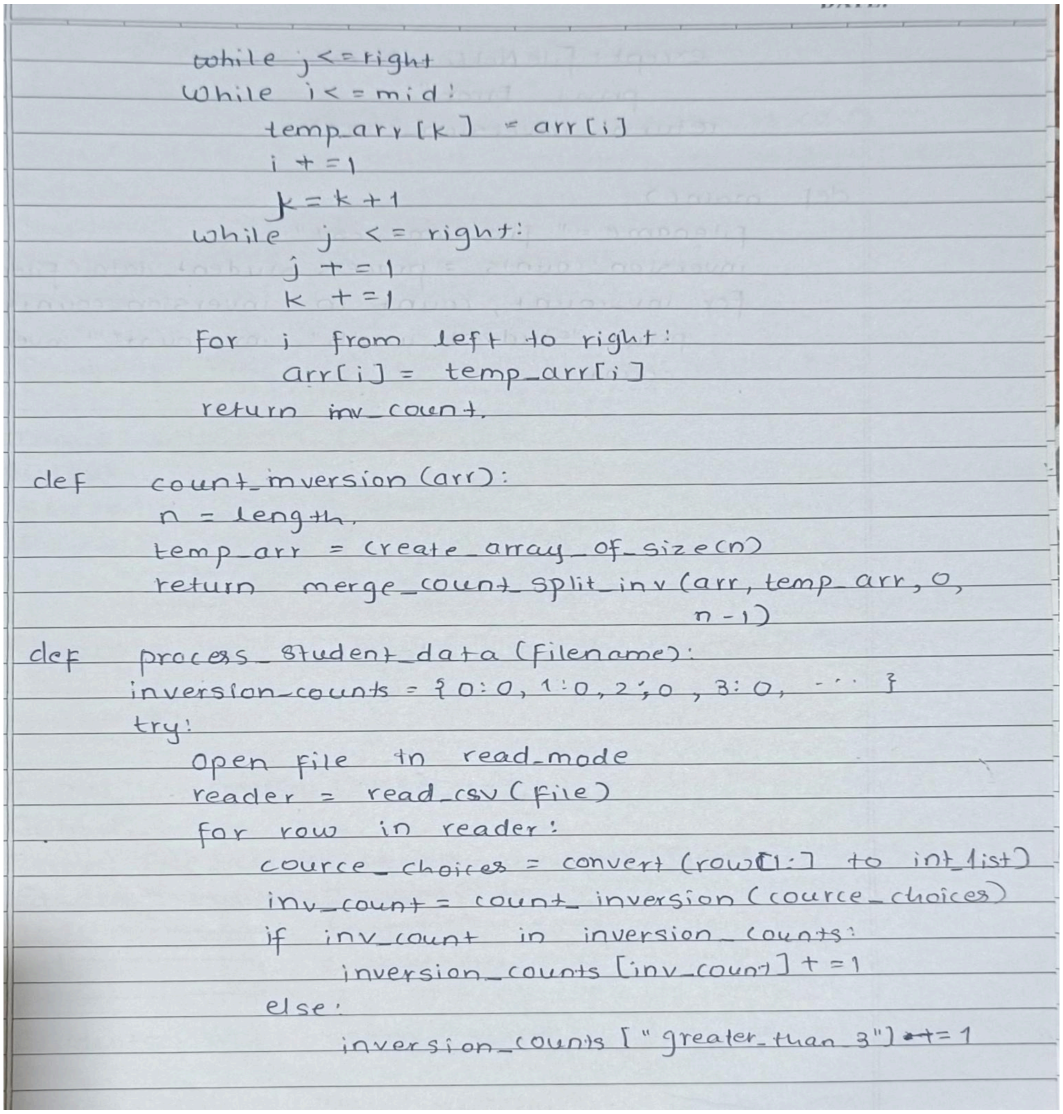
- Compare the elements from left & right subarrays.
- If the element in the left subarray is smaller, add it to sorted array.
- If the element in the right subarray is smaller, count the inversion.
- Add the count inversion ~~in~~ to total count.



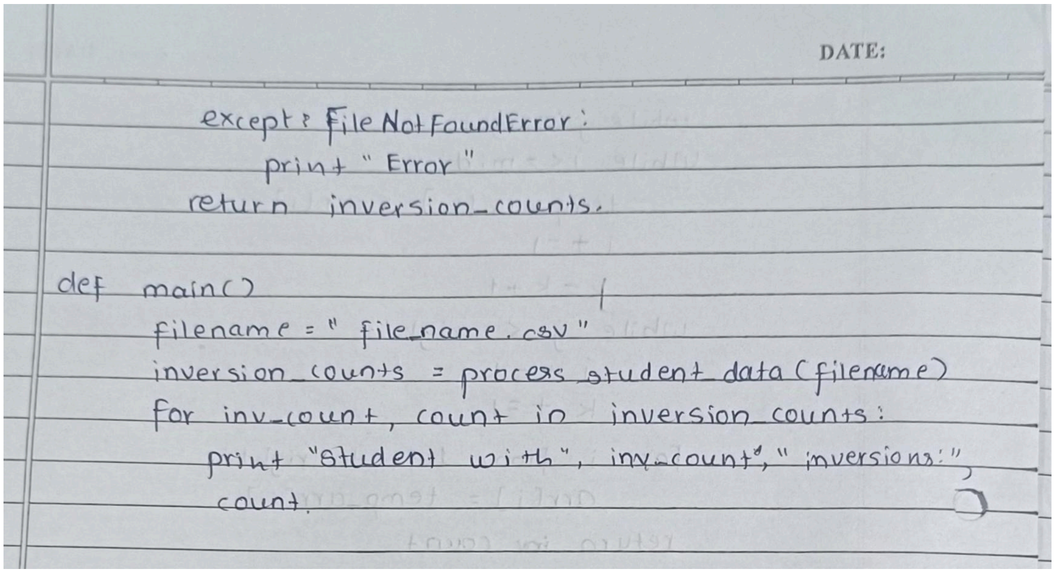
Psuedocode:

```
def  merge_count_split_inv(arr, temp_arr, left, right):
    if  left == right'
            return 0.
    mid = (left +right) // 2
    inv_count = merge_count_split_inv(arr, temp_arr, left, mid)
    inv_count += merge_count_split_inv(arr, temp_arr, mid+1,
                                                  right)
    inv_count += merge_count ~~split_inv~~(arr, temp_arr,
                                        left, mid, right)

    return  inv_count.

def  merge_and_count (arr, temp_arr, left, mid, right):
    i = left
    j = mid +1
    k = left
    inv_count =0
    while  i <= mid  and  j<= right:
        if  arr[i]  <=  arr[j]:
            temp_arr [k] = arr[i]
            i += 1
        else:
            temp_arr [k] = arr[j]
            inv_count  += (mid - i + 1)
            j += 1
        k += 1
```

FOR EDUCATIONAL USE



```
        while j<=right
        While  i<=mid:
            temp_arr [k]  = arr[i]
            i +=1
            k=k+1
        while  j  <= right:
            j +=1
            k  +=1

        For  i  from  left to  right:
            arr[i] =  temp_arr[i]
        return  inv_count.

def     count_inversion (arr):
        n = length.
        temp_arr   = Create_array_of_size(n)
        return   merge_count split_inv (arr, temp_arr, 0,
                                                   n-1)
def     process_student_data (Filename):
       inversion_counts = {0:0, 1:0, 2;0 , 3:0,  ...  }
       try:
           open file   in   read_mode
           reader  =  read_csv (file)
           for  row   in  reader:
               cource_choices = convert (row[1:]  to  int_list)
               inv_count = count_inversion (cource_choices)
               if  inv_count    in   inversion_counts:
                   inversion_counts [inv_count] +=1
               else:
                   inversion_counts ["greater_than_3"] +=1
```



DATE:

```
        except: FileNotFoundError:
            print " Error"
        return  inversion_counts.

def  main()
     filename = " file_name.csv"
     inversion_counts  = process_student_data (filename)
     For inv_count, count in  inversion_counts:
         print "Student with", inv_count, " inversions:",
         count.
```

Code to generate 100 students data:

```
import csv
import random

def generate_student_data(num_students, num_courses, filename):
    """Generate random course choices for students and save to a CSV file."""
    # Open the file in write mode
    with open(filename, mode='w', newline='') as file:
        writer = csv.writer(file)

        # Write the header
        writer.writerow(["StudentID", "Course1", "Course2", "Course3", "Course4"])

        # Generate random course choices for each student
        for student_id in range(1, num_students + 1):
            # Randomly assign courses from the range [1, num_courses]
            courses = random.sample(range(1, num_courses + 1), num_courses)

            # Write the student ID and their course choices
            writer.writerow([student_id] + courses)

# Parameters: 100 students, 4 courses
generate_student_data(100, 4, 'student_course_choices.csv')
```

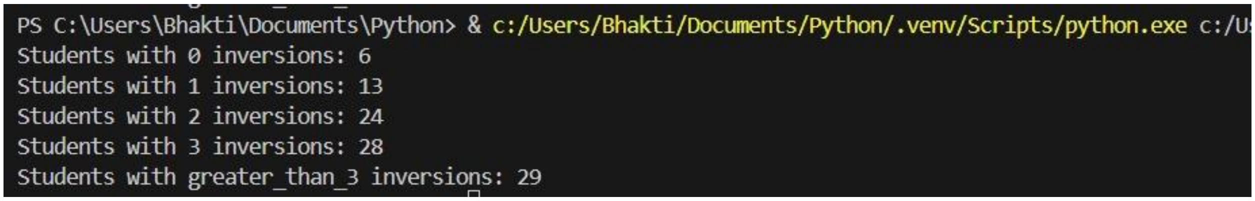
Code:

```
import csv

def merge_count_split_inv(arr, temp_arr, left, right):
    """Merge step that counts the inversions between the two halves."""
```

```python
    if left == right:
        return 0
    mid = (left + right) // 2
    inv_count = merge_count_split_inv(arr, temp_arr, left, mid)
    inv_count += merge_count_split_inv(arr, temp_arr, mid + 1, right)
    inv_count += merge_and_count(arr, temp_arr, left, mid, right)
    return inv_count

def merge_and_count(arr, temp_arr, left, mid, right):
    """Merge the two sorted halves and count inversions."""
    i = left    # Starting index for left subarray
    j = mid + 1 # Starting index for right subarray
    k = left    # Starting index to be sorted
    inv_count = 0

    # Merge the two subarrays
    while i <= mid and j <= right:
        if arr[i] <= arr[j]:
            temp_arr[k] = arr[i]
            i += 1
        else:
            temp_arr[k] = arr[j]
            inv_count += (mid - i + 1)  # All remaining elements in left subarray are greater than arr[j]
            j += 1
        k += 1

    # Copy the remaining elements of left subarray, if any
    while i <= mid:
        temp_arr[k] = arr[i]
        i += 1
        k += 1

    # Copy the remaining elements of right subarray, if any
    while j <= right:
        temp_arr[k] = arr[j]
        j += 1
        k += 1

    # Copy the sorted subarray into the original array
    for i in range(left, right + 1):
        arr[i] = temp_arr[i]

    return inv_count

def count_inversions(arr):
    """Main function to count inversions using divide and conquer (merge sort)."""
    n = len(arr)
    temp_arr = [0] * n
    return merge_count_split_inv(arr, temp_arr, 0, n - 1)

def process_student_data(filename):
    """Process the student data and count inversions for each student."""
    inversion_counts = {0: 0, 1: 0, 2: 0, 3: 0, 'greater_than_3': 0}
    try:
        with open(filename, mode='r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip the header row
```

```python
            for row in reader:
                try:
                    # Convert the course choices to integers
                    course_choices = list(map(int, row[1:]))  # Assuming course codes are in columns 2 onward
                    inv_count = count_inversions(course_choices)
                    # Categorize based on inversion count
                    if inv_count in inversion_counts:
                        inversion_counts[inv_count] += 1
                    else:
                        inversion_counts['greater_than_3'] += 1
                except ValueError:
                    print(f"Error: Invalid data in row {row}")
    except FileNotFoundError:
        print(f"Error: File {filename} not found.")
    return inversion_counts

def main():
    # Filepath to the CSV file containing student data
    filename = 'student_course_choices.csv'
    inversion_counts = process_student_data(filename)

    # Output the inversion counts
    for inv_count, count in inversion_counts.items():
        print(f"Students with {inv_count} inversions: {count}")

if __name__ == "__main__":
    main()
```

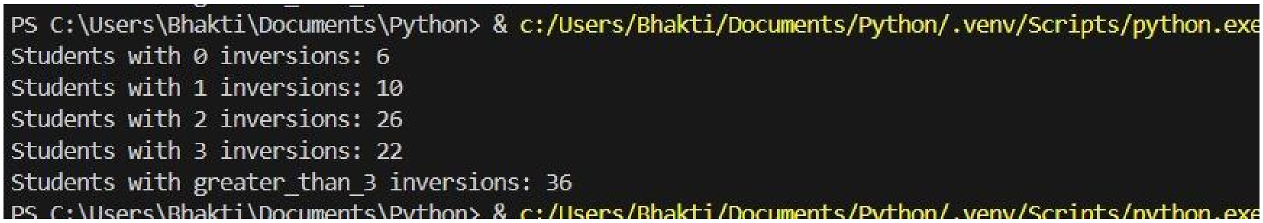
Output:



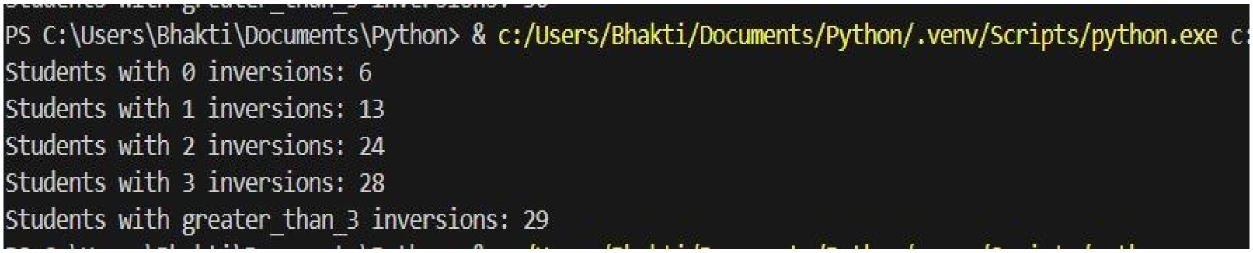
```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe c:/U
Students with 0 inversions: 6
Students with 1 inversions: 13
Students with 2 inversions: 24
Students with 3 inversions: 28
Students with greater_than_3 inversions: 29
```

Test cases:

Positive test case:

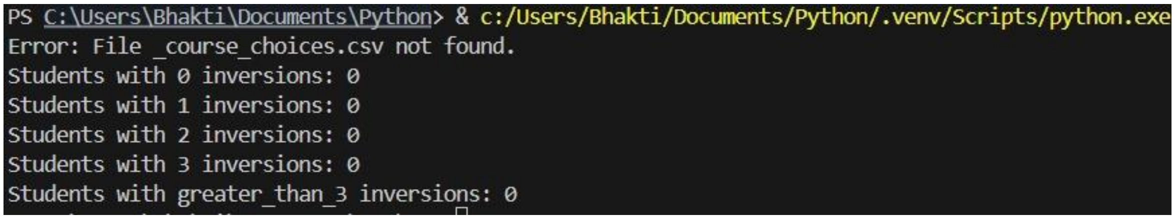


```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe
Students with 0 inversions: 6
Students with 1 inversions: 10
Students with 2 inversions: 26
Students with 3 inversions: 22
Students with greater_than_3 inversions: 36
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe
```

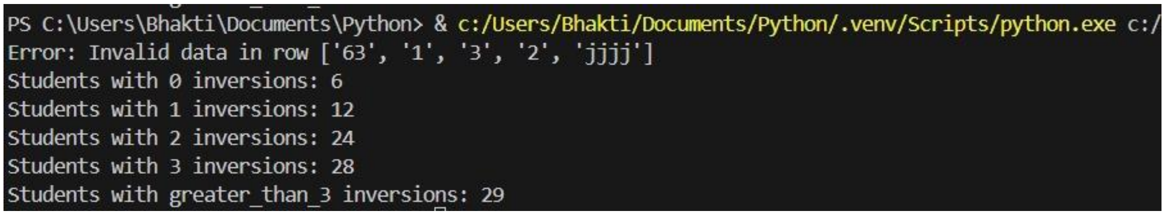


```
Students with greater_than_3 inversions: 36
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe c:
Students with 0 inversions: 6
Students with 1 inversions: 13
Students with 2 inversions: 24
Students with 3 inversions: 28
Students with greater_than_3 inversions: 29
```

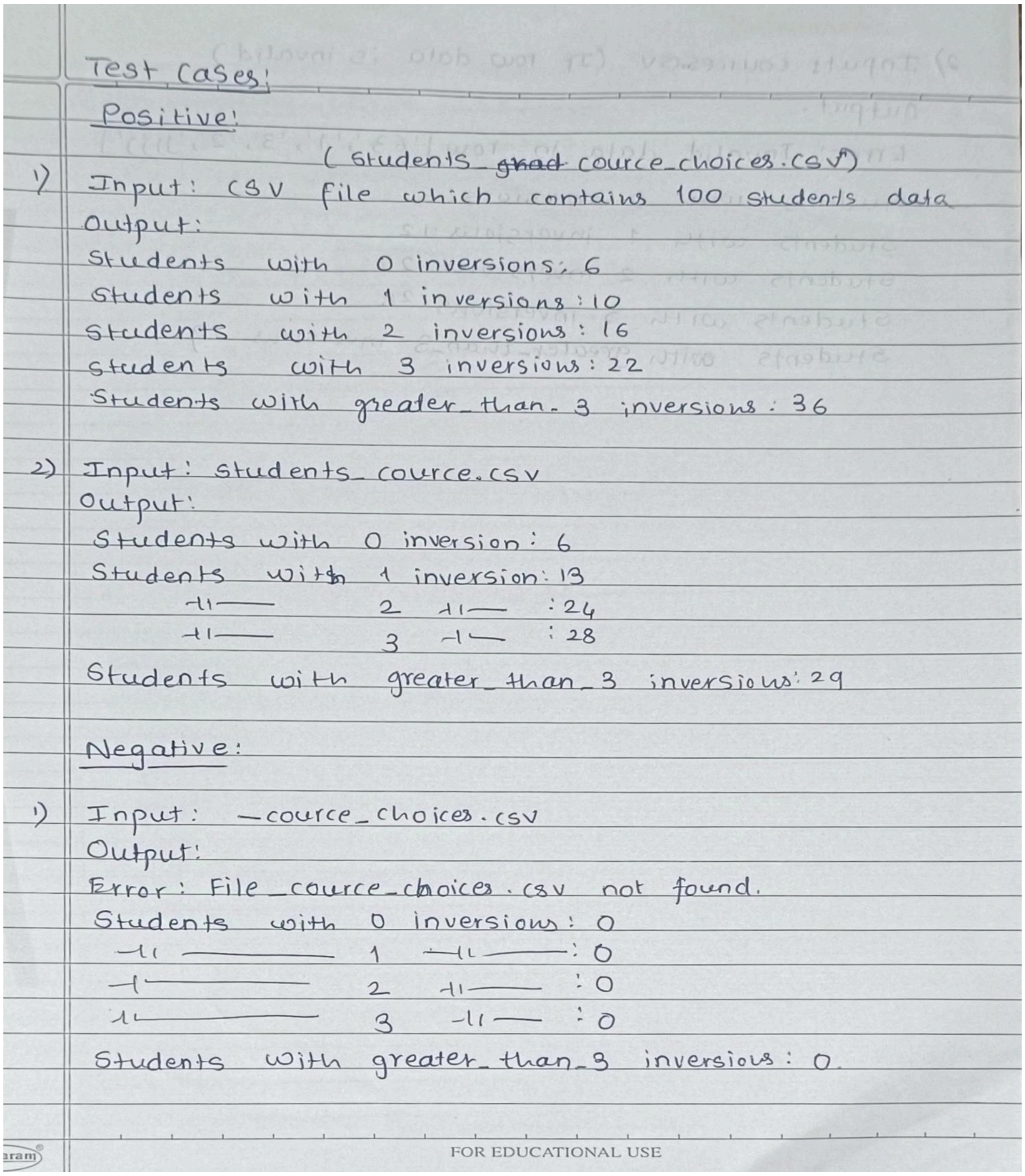
Negative Test Cases:



```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe
Error: File _course_choices.csv not found.
Students with 0 inversions: 0
Students with 1 inversions: 0
Students with 2 inversions: 0
Students with 3 inversions: 0
Students with greater_than_3 inversions: 0
```



```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe c:/
Error: Invalid data in row ['63', '1', '3', '2', 'jjjj']
Students with 0 inversions: 6
Students with 1 inversions: 12
Students with 2 inversions: 24
Students with 3 inversions: 28
Students with greater_than_3 inversions: 29
```



Test Cases:

Positive:

(Students_~~grad~~ cource_choices.csv")

1) Input: CSV file which contains 100 students data

Output:

```
Students with 0 inversions: 6
Students with 1 inversions: 10
Students with 2 inversions: 16
Students with 3 inversions: 22
Students with greater_than-3 inversions: 36
```

2) Input: students_cource.csv

Output:

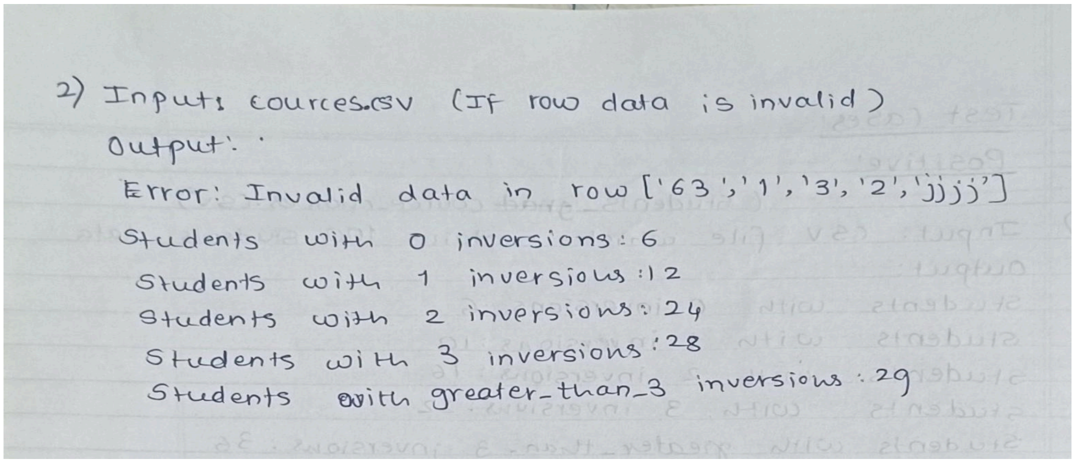
```
Students with 0 inversion: 6
Students with 1 inversion: 13
   -||-       2  -||-    : 24
   -||-       3  -||-    : 28
Students with greater_than_3 inversions: 29
```

Negative:

1) Input: _cource_choices.csv

Output:

```
Error: File_cource_choices.csv not found.
Students with 0 inversions: 0
 -||- ———————— 1 -||- ————: 0
 -||- ———————— 2 -||- ————: 0
 -||- ———————— 3 -||- ————: 0
Students with greater_than_3 inversions: 0.
```

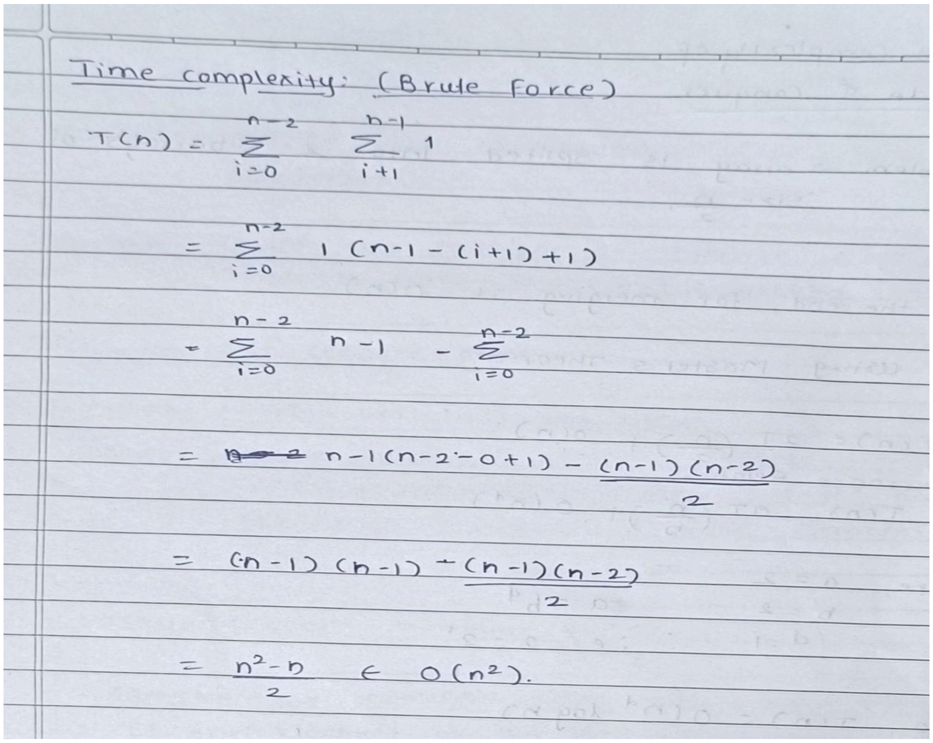


2) Inputs cources.csv (If row data is invalid)

Output:

```
Error: Invalid data in row ['63','1','3','2','jjjj']
Students with 0 inversions: 6
Students with 1 inversions: 12
Students with 2 inversions: 24
Students with 3 inversions: 28
Students with greater_than_3 inversions: 29
```

Time Complexity:



Time complexity: (Brute Force)

$$T(n) = \sum_{i=0}^{n-2} \sum_{i+1}^{n-1} 1$$

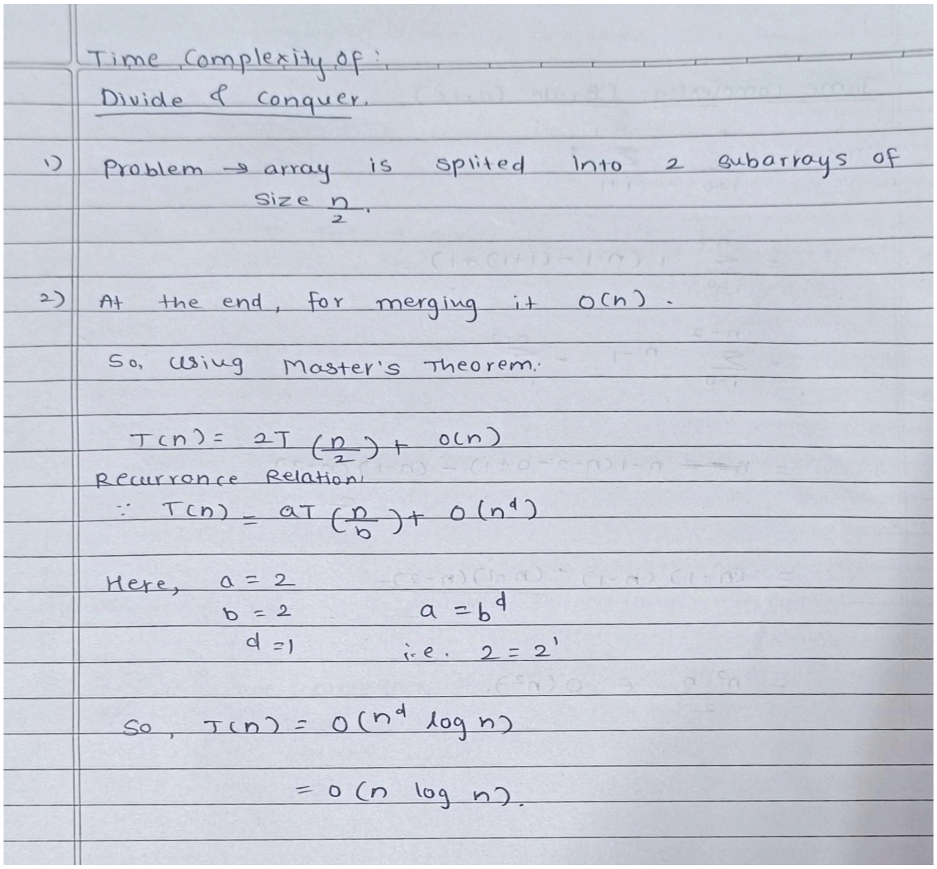
$$= \sum_{i=0}^{n-2} 1\,(n-1-(i+1)+1)$$

$$= \sum_{i=0}^{n-2} n-1 \quad - \sum_{i=0}^{n-2}$$

$$= \cancel{n-2}\; n-1(n-2-0+1) - \frac{(n-1)(n-2)}{2}$$

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

$$= \frac{n^2-n}{2} \in O(n^2).$$



Time Complexity of:
Divide & conquer.

1) Problem → array is splited into 2 subarrays of size $\frac{n}{2}$.

2) At the end, for merging it $O(n)$.

So, using Master's Theorem.

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

Recurrence Relation:

$$\because T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

Here, $a = 2$
$b = 2$ $\qquad a = b^d$
$d = 1$ $\qquad$ i.e. $2 = 2^1$

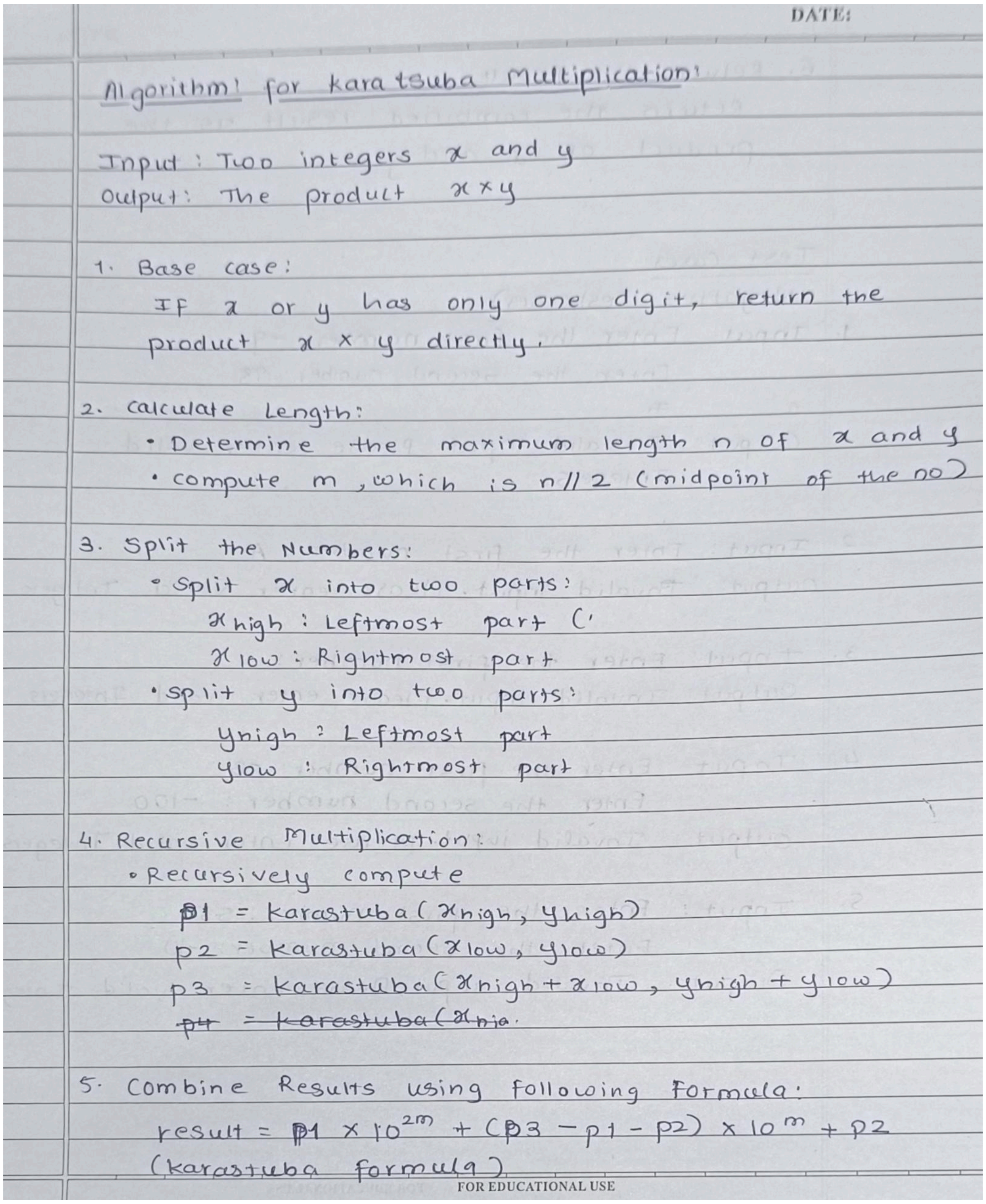
So, $T(n) = O(n^d \log n)$

$= O(n \log n).$

Experiment task 2:

Consider large integers of size 10, 50, 100, 500 and 1000 digits.
Write integer multiplication program
Write an integer multiplication program using divide and conquer technique.

Algorithm:



DATE:

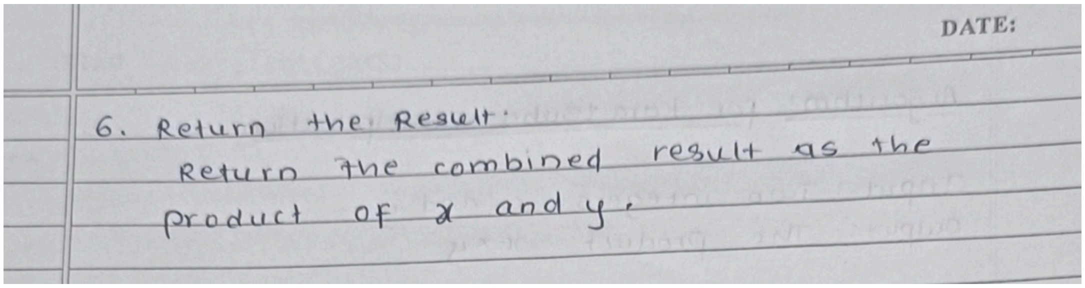
Algorithm! for karatsuba Multiplication!

Input : Two integers x and y
Output: The product $x \times y$

```
1. Base case:
     If x or y has only one digit, return the
     product x × y directly.

2. calculate Length:
     • Determine the maximum length n of x and y
     • compute m, which is n//2 (midpoint of the no)

3. Split the Numbers:
     • Split x into two parts:
         x high : Leftmost part ('
         x low : Rightmost part
     • split y into two parts:
         ynigh : Leftmost part
         ylow : Rightmost part

4. Recursive Multiplication:
     • Recursively compute
         p1 = karastuba(xhigh, yhigh)
         p2 = karastuba(xlow, ylow)
         p3 = karastuba(xhigh + xlow, yhigh + ylow)
         ~~p4 = karastuba(xhig.~~

5. Combine Results using following formula:
     result = $p1 \times 10^{2m} + (p3 - p1 - p2) \times 10^{m} + p2$
     (karastuba formula).
```

FOR EDUCATIONAL USE

:



DATE:

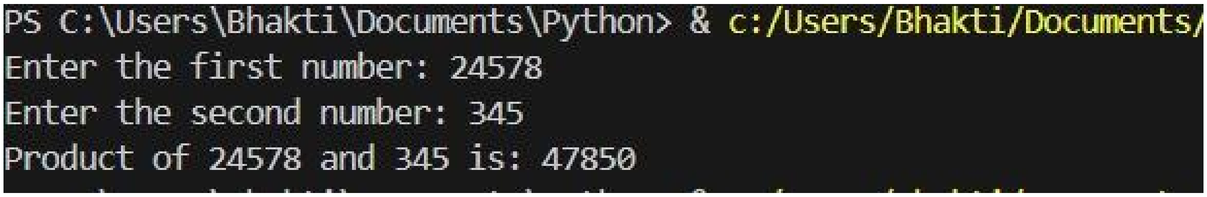
6. Return the Result:

Return the combined result as the product of x and y.

Code:

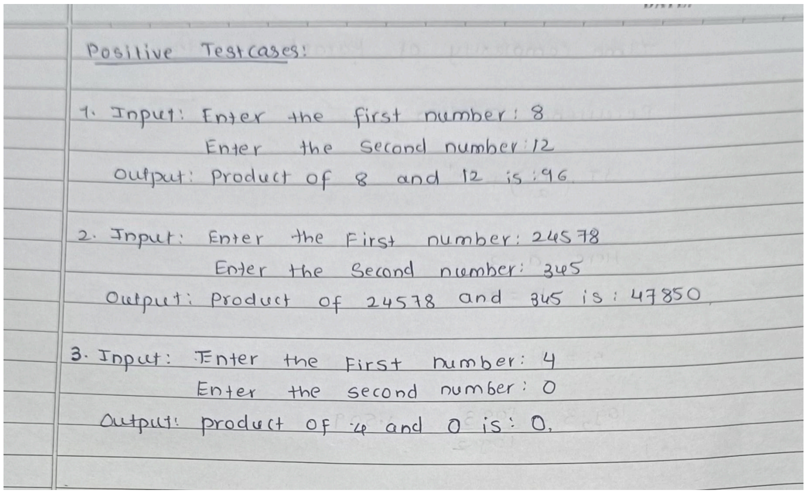
```
def karatsuba(x, y):
    # Convert the numbers to strings to easily split them
    x_str = str(x)
    y_str = str(y)

    # Base case for recursion
    if len(x_str) == 1 or len(y_str) == 1:
        return x * y

    # Length of the numbers
    n = max(len(x_str), len(y_str))
    m = n // 2

    # Split the numbers
    x_high = int(x_str[:-m]) if len(x_str) > m else 0
    x_low = int(x_str[-m:])
    y_high = int(y_str[:-m]) if len(y_str) > m else 0
    y_low = int(y_str[-m:])

    # Recursive calls to calculate the three products
    p1 = karatsuba(x_high, y_high)
    p2 = karatsuba(x_low, y_low)
    p3 = karatsuba(x_high + x_low, y_high + y_low)

    # Combine the results
    return p1 * 10*(2 * m) + (p3 - p1 - p2) * 10*m + p2

# Example usage with user inputs
if __name__ == "__main__":
    try:
        # Taking user inputs
        x = int(input("Enter the first number: "))
        y = int(input("Enter the second number: "))

        # Call the Karatsuba multiplication function
        result = karatsuba(x, y)

        # Print the result
        print(f"Product of {x} and {y} is: {result}")
    except ValueError:
        print("Invalid input. Please enter valid integers.")
```

Output:



```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/
Enter the first number: 24578
Enter the second number: 345
Product of 24578 and 345 is: 47850
```

Test Cases:

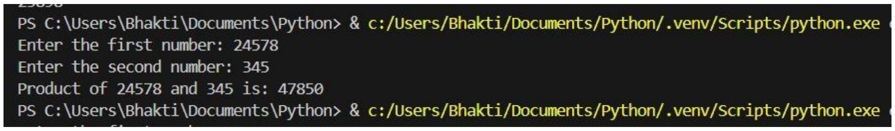


Positive Testcases:

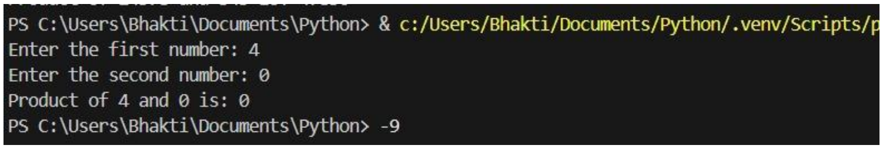
1. Input: Enter the first number: 8
          Enter the Second number: 12
   Output: Product of 8 and 12 is: 96

2. Input: Enter the First number: 24578
          Enter the Second number: 345
   Output: Product of 24578 and 345 is: 47850

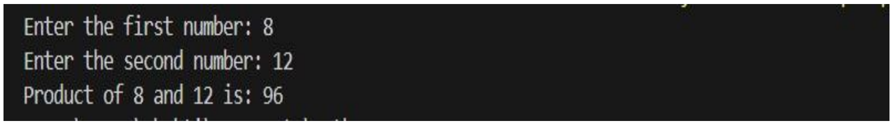
3. Input: Enter the First number: 4
          Enter the second number: 0
   Output: product of 4 and 0 is: 0.



```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe
Enter the first number: 24578
Enter the second number: 345
Product of 24578 and 345 is: 47850
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe
```

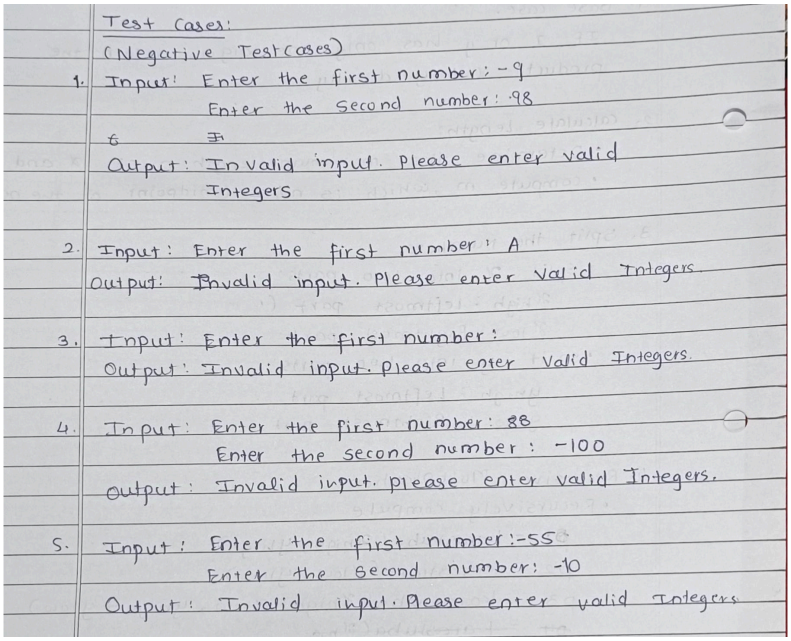


```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/p
Enter the first number: 4
Enter the second number: 0
Product of 4 and 0 is: 0
PS C:\Users\Bhakti\Documents\Python> -9
```



```
Enter the first number: 8
Enter the second number: 12
Product of 8 and 12 is: 96
```

Negative Test Cases:



Test Cases:

(Negative Testcases)

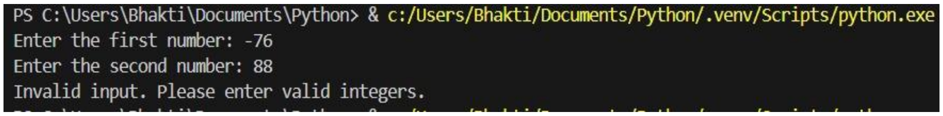
1. Input: Enter the first number: -9
          Enter the Second number: 98
   Output: Invalid input. Please enter valid Integers

2. Input: Enter the first number: A
   Output: Invalid input. Please enter Valid Integers.

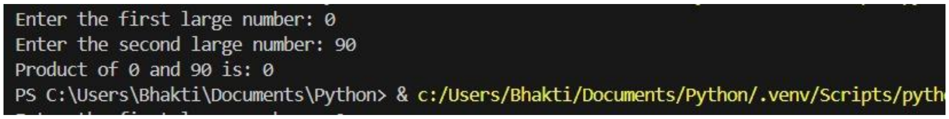
3. Input: Enter the first number:
   Output: Invalid input. Please enter Valid Integers.

4. Input: Enter the first number: 88
          Enter the second number: -100
   Output: Invalid input. please enter valid Integers.

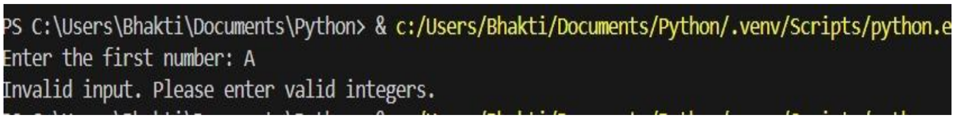
5. Input: Enter the first number: -55
          Enter the second number: -10
   Output: Invalid input. Please enter valid Integers



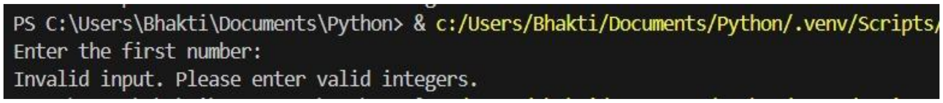
```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.exe
Enter the first number: -76
Enter the second number: 88
Invalid input. Please enter valid integers.
```



```
Enter the first large number: 0
Enter the second large number: 90
Product of 0 and 90 is: 0
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/pyth
```

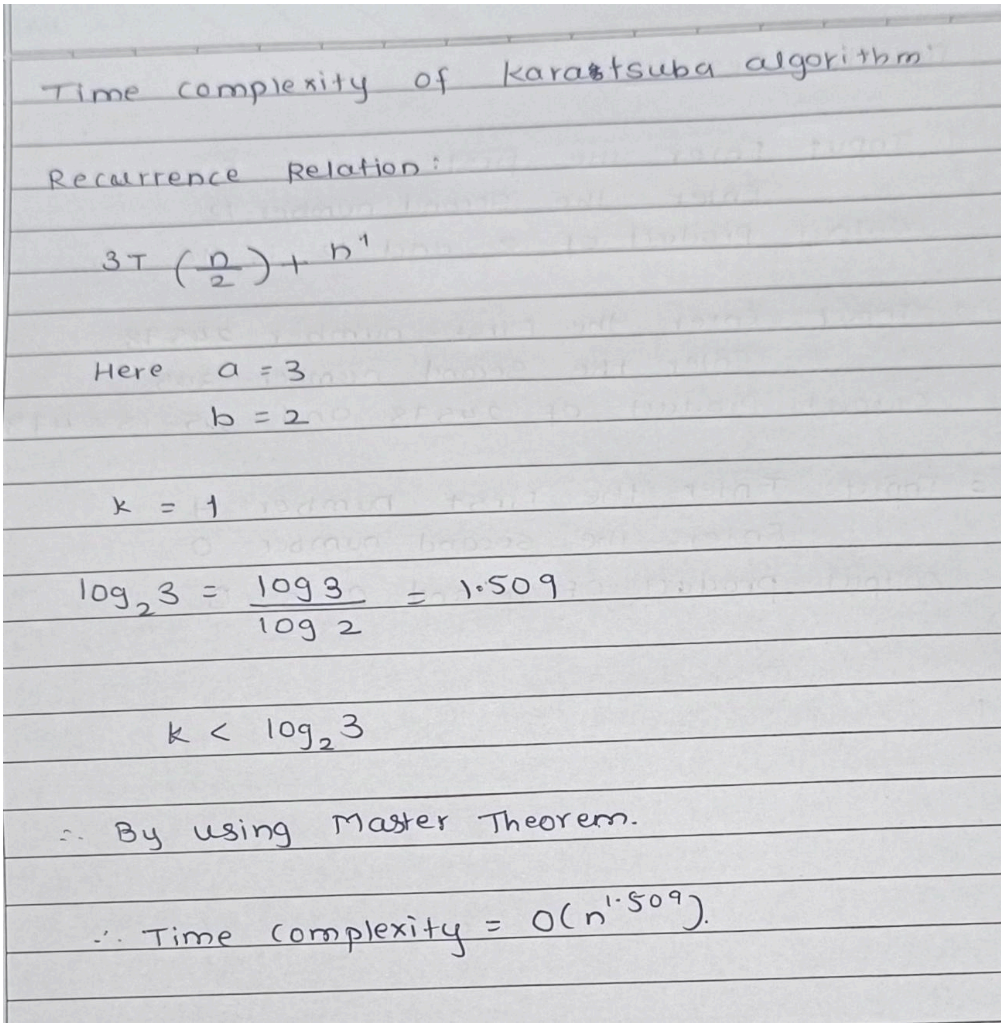


```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/python.e
Enter the first number: A
Invalid input. Please enter valid integers.
```



```
PS C:\Users\Bhakti\Documents\Python> & c:/Users/Bhakti/Documents/Python/.venv/Scripts/
Enter the first number:
Invalid input. Please enter valid integers.
```

Time complexity:



Time complexity of karatsuba algorithm

Recurrence Relation:

$$3T\left(\frac{n}{2}\right) + n^1$$

Here $a = 3$
$b = 2$

$k = 1$

$$\log_2 3 = \frac{\log 3}{\log 2} = 1.509$$

$$k < \log_2 3$$

∴ By using Master Theorem.

∴ Time complexity = $O(n^{1.509})$.

Conclusion:

We learned how to calculate the inversion count of 100 students using divide and conquer algorithm and calculated multiplication of two large numbers using karatsuba multiplication algorithm.