DAA ASSIGNMENT 4

NAME:- Raunak Thanawala

Assignment:- Inversions and Karatsuba

Batch:- CE Group C

AIM:-

Experiment task-1:  
  
Consider first/second year course-code choices of 100 students.   
Find inversion count of these choices.  
Find students with zero, one, two, three inversion counts comment on your result.   
  
  
Experiment task-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.

CODE:-

TASK 1:

1. """
2. Name: Raunak Thanawala
3. Experiment: Inversion Counts
4. Batch: C
5. Registration Number: 231070051
6. """
7. import pandas as pd
8. def process\_student\_course(filepath):
9. """Process the student course data from a CSV file."""
10. data = pd.read\_csv(filepath)
12. if data.empty:
13. raise ValueError("EMPTY FILE")
15. if data.isnull().any().any():
16. raise ValueError("INVALID SIZE OF COLUMNS")
18. *# Validate that all course codes are positive integers*
19. for col in data.columns[2:]:
20. if not data[col].apply(lambda x: isinstance(x, int) and x > 0).all():
21. raise ValueError("INVALID INPUT")
22. def Inversion(filepath):
23. """Count the number of inversions in course codes for each student."""
24. data = pd.read\_csv(filepath)
25. count = {0: 0, 1: 0, 2: 0, 3: 0, 'Other': 0}
27. for index, row in data.iterrows():
28. cc = row[2:].values.tolist()
29. \_, invcount = CountInversions(cc)
31. *# Update the count of inversions*
32. if invcount in count:
33. count[invcount] += 1
34. else:
35. count['Other'] += 1
36. return count
37. def CountInversions(arr):
38. """Recursively count inversions in the array."""
39. if len(arr) <= 1:
40. return arr, 0
42. mid = len(arr) // 2
43. L, left\_inv = CountInversions(arr[:mid])
44. R, right\_inv = CountInversions(arr[mid:])
45. M, split\_inv = MergeAndCountSplit(L, R)
47. total = left\_inv + right\_inv + split\_inv
48. return M, total
49. def MergeAndCountSplit(arr1, arr2):
50. """Merge two sorted arrays and count split inversions."""
51. i = j = split\_inv = 0
52. arr = []
53. n = len(arr1) + len(arr2)
55. for k in range(n):
56. if i >= len(arr1):
57. arr.append(arr2[j])
58. j += 1
59. elif j >= len(arr2):
60. arr.append(arr1[i])
61. i += 1
62. elif arr1[i] <= arr2[j]:
63. arr.append(arr1[i])
64. i += 1
65. else:
66. arr.append(arr2[j])
67. j += 1
68. split\_inv += len(arr1) - i
69. return arr, split\_inv
70. *# Main execution*
71. filepath = "C:/Users/ASUS/Desktop/Coding/DAA/DAA LAB4/student\_course\_codes\_2.csv"
72. process\_student\_course(filepath)
73. result = Inversion(filepath)
74. *# Output the counts of inversions*
75. for invcount, count in result.items():
76. print(f"There are {count} students with {invcount} Inversions")

TASK 2:

def NormalMultiplication(x, y):

    """Perform normal multiplication of two large integers represented as strings."""

    n1 = len(x)

    n2 = len(y)

*# Check for zero inputs*

    if x == "0" or y == "0":

        return "0"

*# Initialize result list*

    prod = ['0'] \* (n1 + n2)

*# Multiply each digit*

    for i in range(n1 - 1, -1, -1):

        carry = 0

        for j in range(n2 - 1, -1, -1):

            product = int(x[i]) \* int(y[j])

            total = product + carry + int(prod[i + j + 1])

            prod[i + j + 1] = str(total % 10)

            carry = total // 10

        prod[i + j] = str(int(prod[i + j]) + carry)

*# Join and remove leading zeros*

    final = ''.join(prod).lstrip('0')

    return final if final else "0"

def Karatsuba(x: str, y: str) -> int:

    """Perform Karatsuba multiplication of two large integers represented as strings."""

    sign1 = -1 if x[0] == '-' else 1

    sign2 = -1 if y[0] == '-' else 1

    x = x.lstrip('-')

    y = y.lstrip('-')

    final\_sign = sign1 \* sign2

    n = max(len(x), len(y))

    x = x.zfill(n)

    y = y.zfill(n)

    if n <= 1:

        return final\_sign \* (int(x) \* int(y))

    mid = n // 2

    x1, x0 = x[:mid], x[mid:]

    y1, y0 = y[:mid], y[mid:]

    p1 = Karatsuba(x1, y1)

    p2 = Karatsuba(x0, y0)

    a = str(int(x1) + int(x0))

    b = str(int(y1) + int(y0))

    ab = Karatsuba(a, b)

    p3 = ab - p1 - p2

    return final\_sign \* (10 \*\* ((n - mid) \* 2) \* p1 + 10 \*\* (n - mid) \* p3 + p2)

def is\_integer(string: str) -> bool:

    """Check if the input string represents a valid integer."""

    if string.startswith('-'):

        return string[1:].isdigit()  *# Check if it's a negative integer*

    return string.isdigit()  *# Check if it's a positive integer*

*# Input and output*

x = input("Enter First Number: ")

y = input("Enter the Second Number: ")

if not is\_integer(x) or not is\_integer(y):

    print("INVALID INPUT")

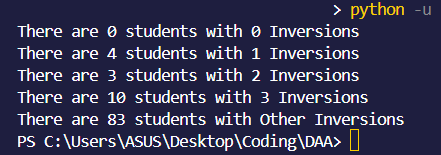
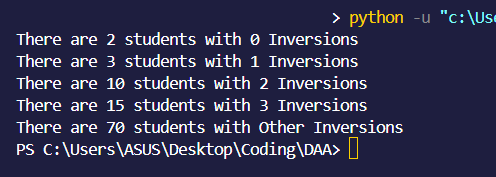
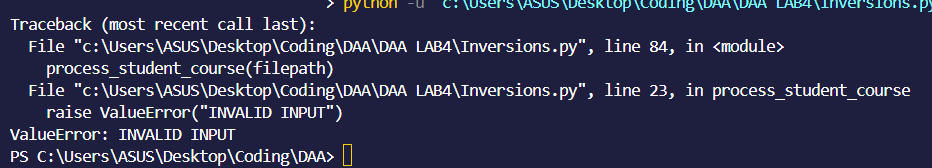
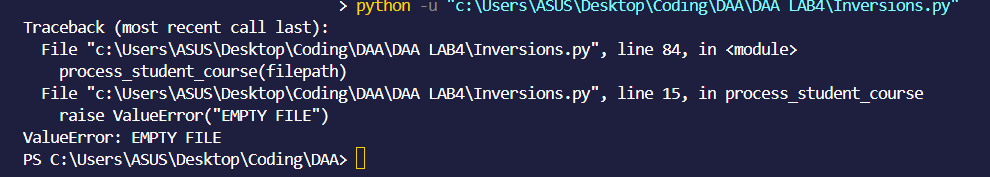
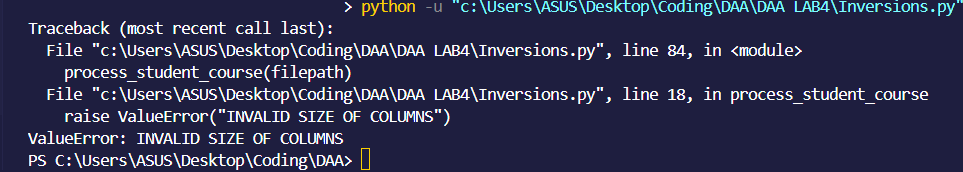
else:

    print(f"Product by Normal Multiplication is {NormalMultiplication(x, y)}")

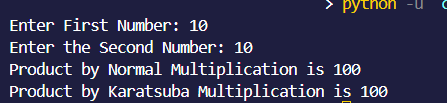
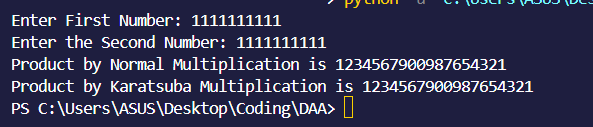
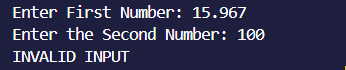
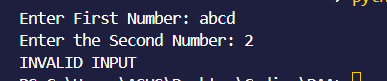
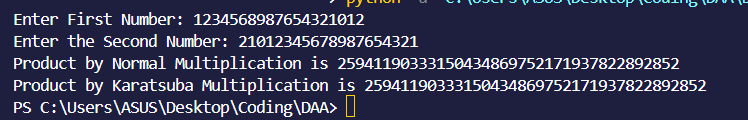
    print(f"Product by Karatsuba Multiplication is {Karatsuba(x, y)}")

TESTCASES IN CODE :-

TASK 1:

1. 
2. 
3. 
4. 
5. 

TASK 2:

1. 
2. 
3. 
4. 
5. 

CONCLUSION :-

So In This Experiment we have Written Pseudo Code Algorithms for finding counting inversions of course codes of students and also we have found how to conduct large integer multiplication by direct and karatsuba’s method.

We have also written 5 Testcases each for each task resulting in total of 10 testcases which shows us the multiple ways we should get an error/ result in our programs.

Then we have written the programs to with the help of our algorithms and testcases eliminating basic errors allowing us to reach our desired goal of 2 programs that give proper errors for invalid inputs and give proper answers that can be proven by calculations

Then we proved that we get the same result by our program as we got by manual calculation from the testcases which shows that our program is 100% correct for our testcases and algorithms.

Then we have also written a way for us to find out how to find the Time Complexities of these Algorithms which shows that some programs even if longer are much shorter to execute such as Karatsuba vs Brute Force Multiplication.