CHAITRA SAMANT

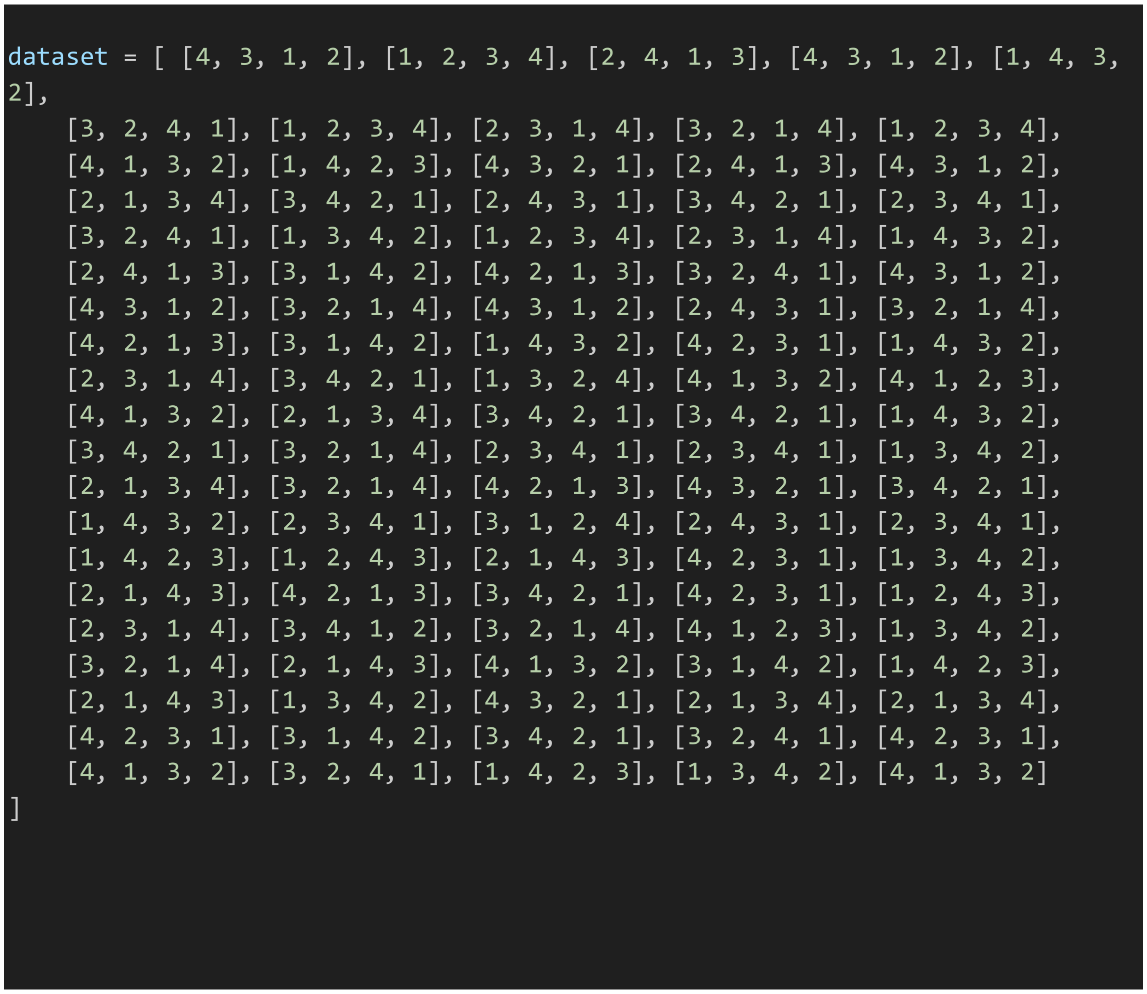
231070055

DAA LAB 04 - A

1. **AIM:**
   To use Divide and Conquer to inversion count in Student Code choices

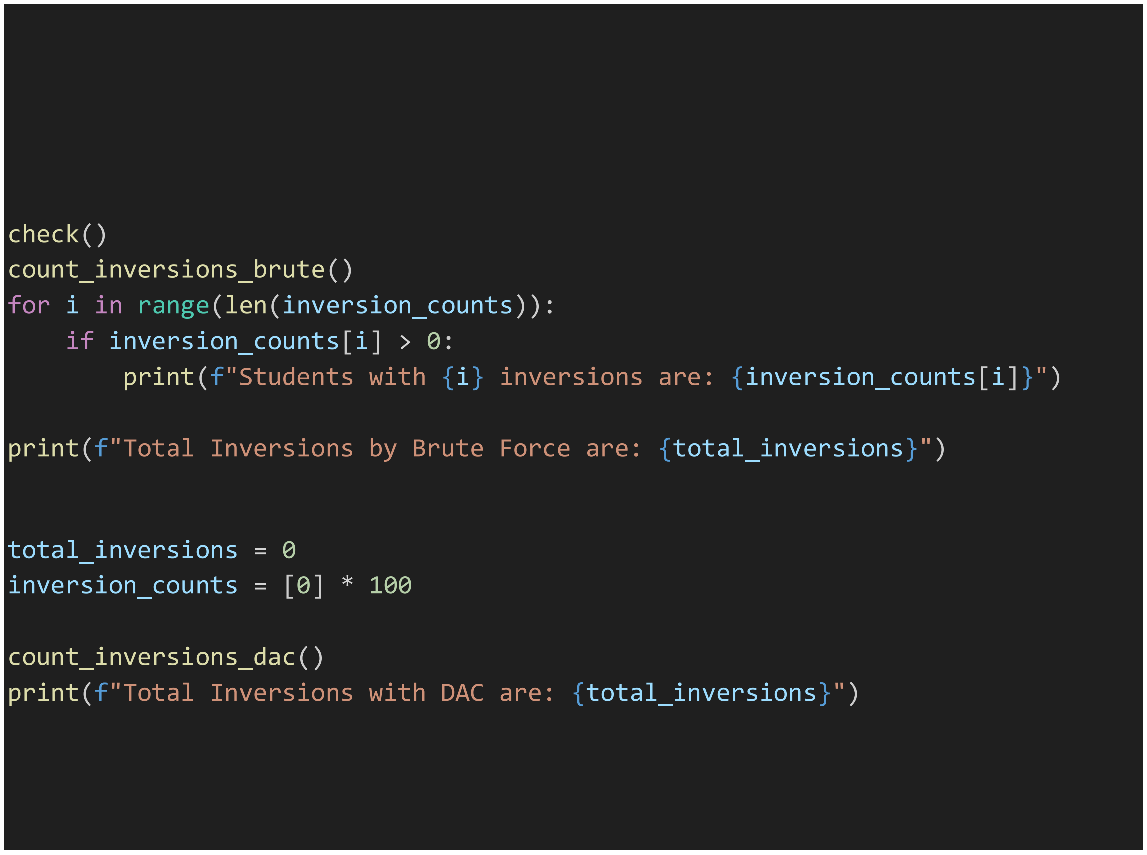
2. **PROGRAM:**

   ASSUMPTION: Here we assume that the course code selected by the users are converted into integer where each course code has its own unique integer. Our dataset consists of these integers instead of the alphanumeric course codes (like R5CE2201T)



```
dataset = [ [4, 3, 1, 2], [1, 2, 3, 4], [2, 4, 1, 3], [4, 3, 1, 2], [1, 4, 3,
2],
    [3, 2, 4, 1], [1, 2, 3, 4], [2, 3, 1, 4], [3, 2, 1, 4], [1, 2, 3, 4],
    [4, 1, 3, 2], [1, 4, 2, 3], [4, 3, 2, 1], [2, 4, 1, 3], [4, 3, 1, 2],
    [2, 1, 3, 4], [3, 4, 2, 1], [2, 4, 3, 1], [3, 4, 2, 1], [2, 3, 4, 1],
    [3, 2, 4, 1], [1, 3, 4, 2], [1, 2, 3, 4], [2, 3, 1, 4], [1, 4, 3, 2],
    [2, 4, 1, 3], [3, 1, 4, 2], [4, 2, 1, 3], [3, 2, 4, 1], [4, 3, 1, 2],
    [4, 3, 1, 2], [3, 2, 1, 4], [4, 3, 1, 2], [2, 4, 3, 1], [3, 2, 1, 4],
    [4, 2, 1, 3], [3, 1, 4, 2], [1, 4, 3, 2], [4, 2, 3, 1], [1, 4, 3, 2],
    [2, 3, 1, 4], [3, 4, 2, 1], [1, 3, 2, 4], [4, 1, 3, 2], [4, 1, 2, 3],
    [4, 1, 3, 2], [2, 1, 3, 4], [3, 4, 2, 1], [3, 4, 2, 1], [1, 4, 3, 2],
    [3, 4, 2, 1], [3, 2, 1, 4], [2, 3, 4, 1], [2, 3, 4, 1], [1, 3, 4, 2],
    [2, 1, 3, 4], [3, 2, 1, 4], [4, 2, 1, 3], [4, 3, 2, 1], [3, 4, 2, 1],
    [1, 4, 3, 2], [2, 3, 4, 1], [3, 1, 2, 4], [2, 4, 3, 1], [2, 3, 4, 1],
    [1, 4, 2, 3], [1, 2, 4, 3], [2, 1, 4, 3], [4, 2, 3, 1], [1, 3, 4, 2],
    [2, 1, 4, 3], [4, 2, 1, 3], [3, 4, 2, 1], [4, 2, 3, 1], [1, 2, 4, 3],
    [2, 3, 1, 4], [3, 4, 1, 2], [3, 2, 1, 4], [4, 1, 2, 3], [1, 3, 4, 2],
    [3, 2, 1, 4], [2, 1, 4, 3], [4, 1, 3, 2], [3, 1, 4, 2], [1, 4, 2, 3],
    [2, 1, 4, 3], [1, 3, 4, 2], [4, 3, 2, 1], [2, 1, 3, 4], [2, 1, 3, 4],
    [4, 2, 3, 1], [3, 1, 4, 2], [3, 4, 2, 1], [3, 2, 4, 1], [4, 2, 3, 1],
    [4, 1, 3, 2], [3, 2, 4, 1], [1, 4, 2, 3], [1, 3, 4, 2], [4, 1, 3, 2]
]
```

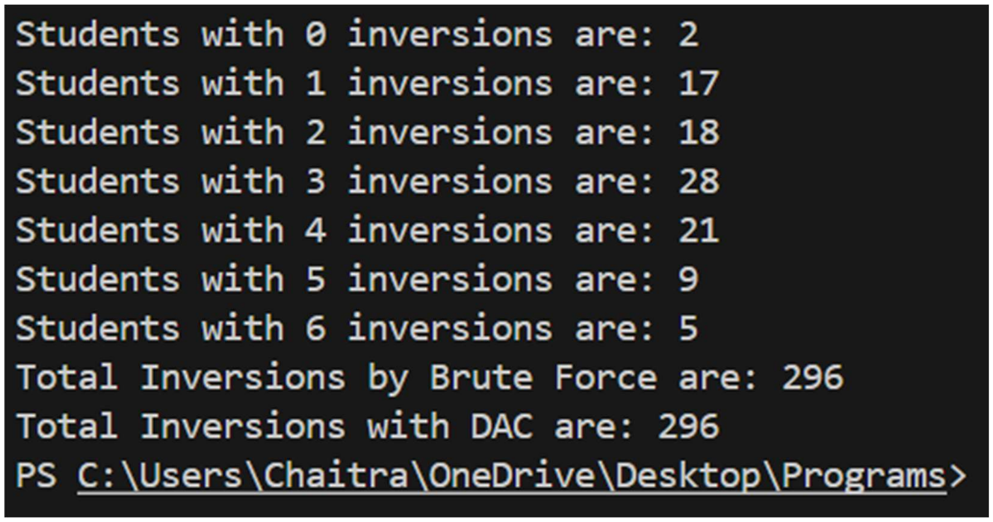
```python
def check():
    """
    This function checks whether the input dataset is valid

    Arguments: None, it uses global variable dataset

    Returns:
    Error message if it exists and terminates the program

    """
    if len(dataset) == 0:
        print("Dataset is empty")
        exit()
    for s in dataset:
        for i in s:
            if i<0:
                print("Dataset can't have negative values")
                exit()

            if int(i)!=i:
                print("Course codes have to be integers")
                exit()



        if s ==[0]*len(s):
            print("Course choices cannot be 0")
            exit()



total_inversions = 0
inversion_counts = [0] * 100

def count_inversions_brute():
    """
    This function uses brute force using nested loops to calculate inversion
count

    Arguments:
    None, it uses global variable dataset

    Returns:
    total_inversions (int) : Integer value of total number of count inversions

    """
    global total_inversions
    for stud in dataset:
        count = 0
        n = len(stud)

        for i in range(n):
            for j in range(i + 1, n):
                if stud[i] > stud[j]:
                    count += 1

        total_inversions += count
        inversion_counts[count] += 1


def merge(arr, temp_arr, left, mid, right):
    """
    This function merges left and right subarray in combine step

    Arguments:
    arr (list): The array in which count needs to be found
    temp_arr (list): This array temporarily stores sorted values of subarray
    left (int): This is the left index of the subarray
    right (int): This is the right index of the subarray
    mid (int): This is the middle index of the subarray

    Returns:
    tot_inversions (int): Integer value of total inversions

    """
    global total_inversions
    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:
            inv_count += (mid - i + 1)
            temp_arr[k] = arr[j]
            j += 1
        k += 1

    while i <= mid:
        temp_arr[k] = arr[i]
        i += 1
        k += 1

    while j <= right:
        temp_arr[k] = arr[j]
        j += 1
        k += 1

    for i in range(left, right + 1):
        arr[i] = temp_arr[i]

    return inv_count

def ms(arr, temp_arr, left, right):
    """
    This function recursively implements merge sort algorithm

    Arguments:
    arr (list): Array in which merge sort needs to be implemented
    left(int): Value of the left index
    right(int): Value of the right index

    Returns:
    inv_count (int): Total inversion count in the array

    """
    inv_count = 0
    if left < right:
        mid = (left + right) // 2

        inv_count += ms(arr, temp_arr, left, mid)
        inv_count += ms(arr, temp_arr, mid + 1, right)
        inv_count += merge(arr, temp_arr, left, mid, right)

    return inv_count

def count_inversions_dac():
    """
    This function uses DAC using merge sort to calculate inversion count

    Arguments:
    None, it uses global variable dataset

    Returns:
    total_inversions (int) : Integer value of total number of count inversions

    """
    global total_inversions
    for s in dataset:
        n = len(s)
        temp_arr = [0] * n
        total_inversions += ms(s, temp_arr, 0, n - 1)
```



```python
check()
count_inversions_brute()
for i in range(len(inversion_counts)):
    if inversion_counts[i] > 0:
        print(f"Students with {i} inversions are: {inversion_counts[i]}")

print(f"Total Inversions by Brute Force are: {total_inversions}")


total_inversions = 0
inversion_counts = [0] * 100

count_inversions_dac()
print(f"Total Inversions with DAC are: {total_inversions}")
```

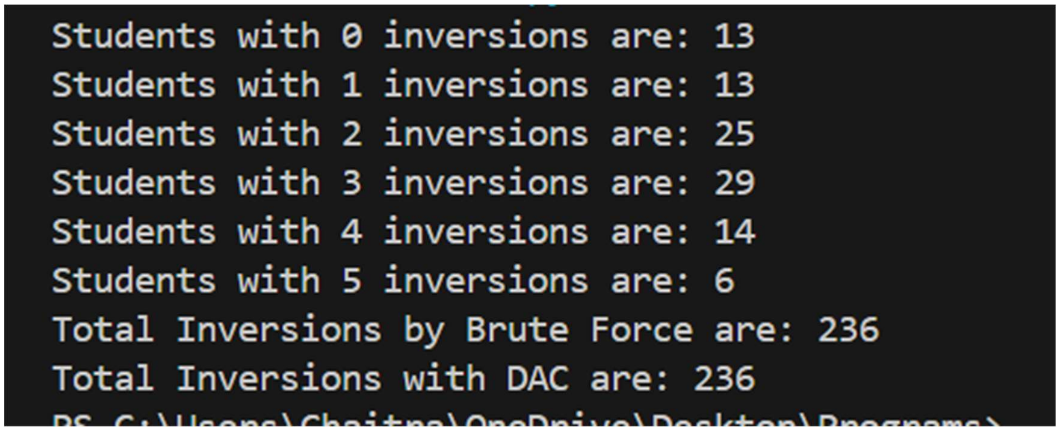
3. **TESTCASES:**

- POSITIVE
   A. Dataset 1



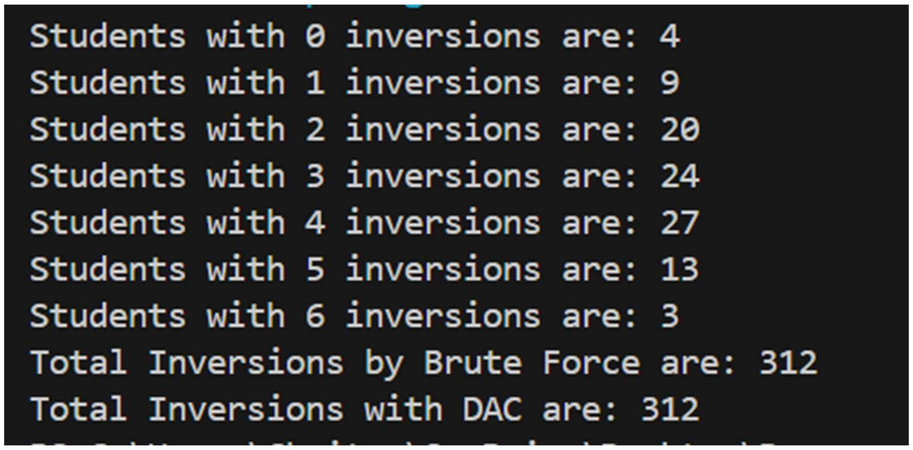
```
Students with 0 inversions are: 2
Students with 1 inversions are: 17
Students with 2 inversions are: 18
Students with 3 inversions are: 28
Students with 4 inversions are: 21
Students with 5 inversions are: 9
Students with 6 inversions are: 5
Total Inversions by Brute Force are: 296
Total Inversions with DAC are: 296
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

   B. Dataset 2



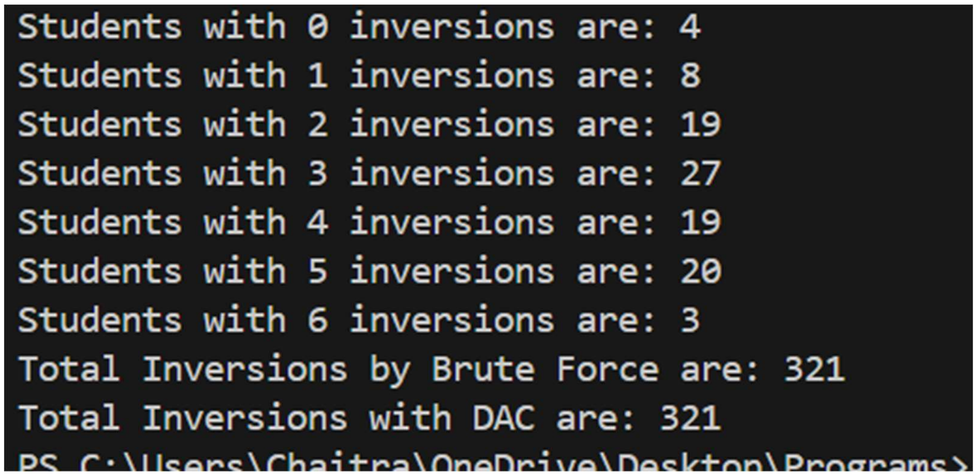
```
Students with 0 inversions are: 13
Students with 1 inversions are: 13
Students with 2 inversions are: 25
Students with 3 inversions are: 29
Students with 4 inversions are: 14
Students with 5 inversions are: 6
Total Inversions by Brute Force are: 236
Total Inversions with DAC are: 236
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

   C. Dataset 3



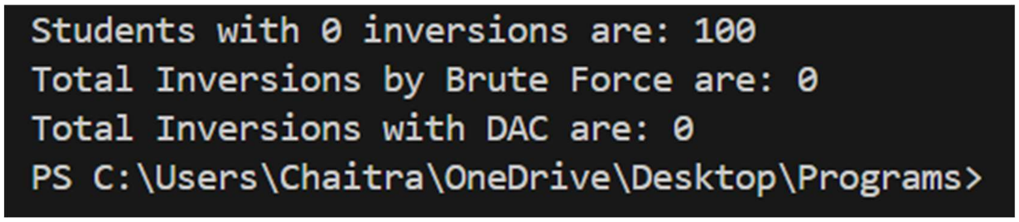
```
Students with 0 inversions are: 4
Students with 1 inversions are: 9
Students with 2 inversions are: 20
Students with 3 inversions are: 24
Students with 4 inversions are: 27
Students with 5 inversions are: 13
Students with 6 inversions are: 3
Total Inversions by Brute Force are: 312
Total Inversions with DAC are: 312
```

D. Dataset 4



```
Students with 0 inversions are: 4
Students with 1 inversions are: 8
Students with 2 inversions are: 19
Students with 3 inversions are: 27
Students with 4 inversions are: 19
Students with 5 inversions are: 20
Students with 6 inversions are: 3
Total Inversions by Brute Force are: 321
Total Inversions with DAC are: 321
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

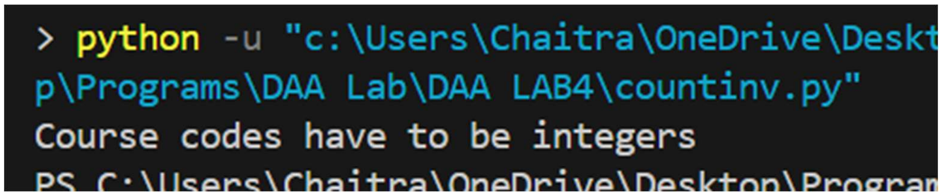
E. Dataset 5 (all students choose ideal courses)



```
Students with 0 inversions are: 100
Total Inversions by Brute Force are: 0
Total Inversions with DAC are: 0
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

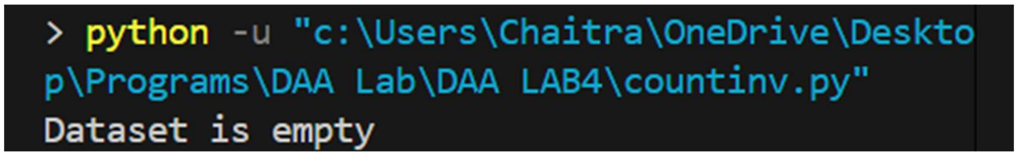
- NEGATIVE

A. Array with non integer values (dataset = [[1.1, 3.2, 2.0, 4]])



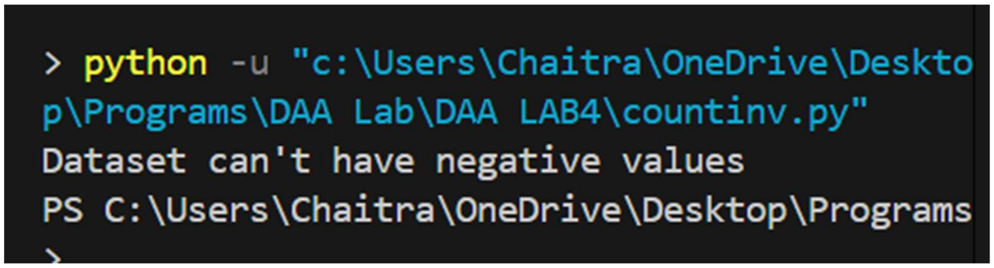
```
> python -u "c:\Users\Chaitra\OneDrive\Deskt
p\Programs\DAA Lab\DAA LAB4\countinv.py"
Course codes have to be integers
PS C:\Users\Chaitra\OneDrive\Desktop\Progran
```

B. Array is empty (dataset = [])



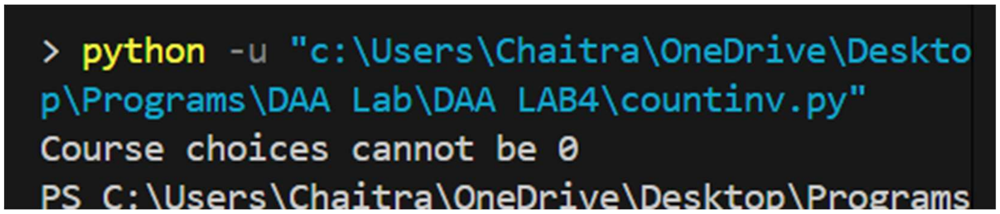
```
> python -u "c:\Users\Chaitra\OneDrive\Deskto
p\Programs\DAA Lab\DAA LAB4\countinv.py"
Dataset is empty
```

C. Negative Values of choices (dataset = [[-1,2,3,4]])



```
> python -u "c:\Users\Chaitra\OneDrive\Deskto
p\Programs\DAA Lab\DAA LAB4\countinv.py"
Dataset can't have negative values
PS C:\Users\Chaitra\OneDrive\Desktop\Programs
>
```

D. Zero as the only value of choices (dataset = [[0,0,0,0]])



```
> python -u "c:\Users\Chaitra\OneDrive\Deskto
p\Programs\DAA Lab\DAA LAB4\countinv.py"
Course choices cannot be 0
PS C:\Users\Chaitra\OneDrive\Desktop\Programs
```

4. **CONCLUSION:**

Students with 0 Inversions: These students are choosing courses in an ideal order, in the order the courses were expected to be selected.

Students with 1 inversion: These students are choosing one course in a different order than expected, this can still be acceptable as there could be various reason for a student choosing one course earlier.

Student with 2+ inversions: These students are choosing multiple courses not in the correct order. This is alarming and we must understand why a particular student is not choosing the courses in order as they were expected to be chosen.

Thus, we calculated count inversions of 100 students and their choices made while choosing the courses. We also understood what can be interpreted from the results. We used brute force and Merge Sort (DAC) algorithms to calculate inversion count