(Practice 22)

1. Read file marks.txt (already shared in last revision class). Find and print the student with minimum average and maximum average.
2. Read file marks.txt (already shared in last revision class). Create another file "marks1.txt". Read marks and write in file by writing marks of each class in a single line. File output should be like:

```
3
4
25 78 36 64
3
66 79 58
2
75 64
```

3. Write code to create a file similar to marks.txt. The file should contain marks of 5 classes. Each class has marks of 7 to 10 students and each student has marks 0-100 at random.
4. Write code to create a file matrix.txt. Generate random number for rows (4-8) and columns (4-8). Write rows and columns on the top of the file, each in single line. Next in rows * columns lines, write random elements of the matrix in range 1 to 9.

(Practice 23)

5. Create two files with the names "matrix_set1" and "matrix_set2". Both files will contain 10 matrices of different dimensions. However, the corresponding matrices in both files should have same dimensions. For example, if matrix 2 in file 1 has dimensions 3 x 5, then matrix 2 in file 2 should have dimensions 3 x 5.
6. Write code to create these file. The matrix dimension can vary from 2 to 6, where rows and columns can be different or same. Write elements of each matrix in random 0 to 9 both including.

Sample File 1:

```
10          There are 10 matrices
2           First matrix has 2 rows
3           First matrix has 3 columns
2
7
3
6
9
3
3           Second matrix has 3 rows
...
```

7. Read two files created in task 01 and create two more matrices with the names "sum.txt" and "diif.txt". Read previous two matrices and store their sum in "sum.txt" and difference in file "diff.txt".

```
10
2
3
...
```

8. Consider file "marks.txt" for next tasks. The file contains marks of 5000 students. Some students fail to appear in the exam. If you open file in note pad and search -2, you will find some values just next to the roll no, showing that these students have not appeared in the exam.

Another thing is some students fail to appear in some subject only. The value -1 indicates that student has not given the exam of the subject. You are required to do following tasks:

9. Read the file and count how many students have given the exam
10. Input roll no, search if student has failed to appear in the exam, print appropriate message. Otherwise, print the marks of the student. If he/ she has not appeared in some subject, give message in front of the subject. Also, calculate the average marks of the student (calculate average out of the subjects, student has appeared)
11. Read the file and count and print how many students have appeared in all subjects
12. Read the file and count how many students have appeared in all subjects. Open another file for writing with name "marks_appear.txt". Write the count of appeared students on top of the file. Reopen the previous file and copy roll no and marks of the students in new file, for students who have appeared in all the exams
13. Read the file and count how many students have not appeared in the exams. Open another file for writing with name "absent.txt". Write the count of absent students on top of the file. Next, write only the roll nos of students, who have not appeared in the exams
14. Read the file and count students who have not appeared in all exams. Open another file for writing with name "marks_absent.txt". Write the count of such students on top of the file. Next, write roll nos and marks of students who have not appeared in all the exams

(Practice 25)

15. Declare a list of 10 elements. Initialize elements at random (with any range of your choice). Print elements in single line. Find and print average. Subtract all element from average. Print elements again in single line. Next, count and print number of negative elements and number of positive elements.

Sample Runs:

```
Length: 10
37 17 61 95 51 72 12 49 80 92
Average: 56.60
-19.60 -39.60 4.40 38.40 -5.60 15.40 -44.60 -7.60 23.40 35.40
Count of positive values: 5
Count of negative values: 5
Length: 10
18 96 79 13 54 39 54 28 67 86
Average: 53.40
-35.40 42.60 25.60 -40.40 0.60 -14.40 0.60 -25.40 13.60 32.60
Count of positive values: 6
Count of negative values: 4
```

16. Declare a list of 30 elements to store marks of 30 students of a class. Initialize elements at random in range 1-100. The management is interested to divide students in three categories i.e. failures, above average and below average. For this purpose, calculate average of passed students only. Do following tasks in steps:

- Initialize marks and print in single line
- Print marks in a single line
- count number of pass students (students with 50 or more marks are pass)
- sum number of pass students

- calculate and print average of pass students

- Next, print marks of fail students in single line
- Next, print marks of students with above average

- Lastly, print marks of student with below average

Note: Run different loop for each step

Sample Runs:

```
80 66 97 69 90 77 92 45 14 50 5 79 73 94 62 91 18 89 63 35 11 10 49 1 46 52 82
41 35 99
Average: 78.06
45 14 5 18 35 11 10 49 1 46 41 35
80 97 90 92 79 94 91 89 82 99
66 69 77 45 14 50 5 73 62 18 63 35 11 10 49 1 46 52 41 35
```

17. Declare a list of 12 elements to store monthly sales of XYZ Company. Initialize elements at random (range 1000 to 2000) and print in single line. Company want to do different analysis. For example, difference of sale in first half of the year and second half of the year. Difference of sales in each quarter.

A year has two halves. First half has first six month and second half has next six month. Print total sales in first & second half.

18. There are four quarters of three months in a year. Like first quarter has January, February & March. The second quarter has April, May & June. Print sales of each quarter. (By adding sales of consecutive three months)

Note: Run different loop for each step

Sample Runs:

```
1081 1380 1789 1037 1865 1703 1549 1694 1526 1768 1557 1746
Sale in Two Halves
First Half: 8855
Second Half: 9840
Quarter Wise Sale
Sale in Quarter 1: 4250
Sale in Quarter 2: 4605
Sale in Quarter 3: 4769
Sale in Quarter 4: 5071
```

(Practice 26)

Use following code in main for following tasks:

```
int main(){
      int length, i;
      srand(time(0));
      length = rand() % 11 + 5;
      int x [length];
      for (i = 0; i < length ; i++){
            x[i] = rand() % 101;
            printf ("%d ", x[i]);
      printf("\n");
```

```
from random import *
length = randint(5, 15)
x = [0] * length
for i in range(length)
      x[i]] = randint(0, 100)
      print (x[i], end = ' ')
print()
```

19. Write code to print adjacent pairs, which are in order (means first element is smaller than equal second element)

Sample Run:

```
23 45 18 17 36
Pairs in order:
23, 45
17, 36
```

20. Create another list of same size with random numbers. Compare corresponding elements of both lists and for each pair print larger element.

Sample Run:
```
List 1: 23 45 18 17 36
List 2: 41 14 11 37 46
41 45 18 37 46
```

21. Extend previous task. Compare corresponding elements of both lists and put smaller element in list 1 and larger elements in list 2 and print both lists again.

Sample Run:
```
List 1: 23 45 18 17 36
List 2: 41 14 11 37 46

Smaller: 23 14 11 17 36
Larger: 41 45 18 37 46
```

22. For the list check and print, whether list is sorted or not. Note, you don't have to sort, just check adjacent pairs, if all adjacent pairs are in order, the list will be in order.

Sample Run:
```
23 45 18 17 36
List not sorted
12 23 35 48 57 76
List is sorted
```

(Practice 27)

Consider array/ list having duplicate elements for task 1-3.

```
x = [23, 45, 18, 23, 17, 45, 36, 23, 45, 18, 36, 45, 18, 17, 36, 23, 17]

int x[] = {23, 45, 18, 23, 17, 45, 36, 23, 45, 18, 36, 45, 18, 17, 36, 23, 17};
```

23. Write code to print distinct elements only:

Sample Run:
```
23 45 18 23 17 45 36 23 45 18 36 45 18 17 36 23 17
23 45 18 17 36
```

24. Create another array/ list having distinct elements only:

Sample Run:
```
X: 23 45 18 23 17 45 23 45 18 36 45 18 17 36 23 17
Y: 23 45 18 17 36
```

25. Create two arrays/ lists having distinct elements and their counts/ frequency:

Sample Run:
```
X: 23 45 18 23 17 45 36 23 45 18 36 45 18 17 23 17
Y: 23 45 18 17 36
C:  4  4  3  3  2
```

26. Create a 2D list of size 4 x 3. Means, there are four rows and three columns. Initialize elements at random with two-digit numbers. Print elements in single line. Next, print elements in tabular form. Next, print element in form, where columns are printed in rows ad rows are printed in columns:

Sample Run:
```
31 42 73 24 15 96 78 44 62 20 39 58

31 42 73
24 15 96
78 44 62
```

20 39 58

31 24 78 20
42 15 44 39
73 96 62 58

27. Create a 2D list of size 4 x 4. Means, there are four rows and four columns. Initialize elements at random with two-digit numbers. Print elements in single line. Next, print both diagonals in separate lines.

Sample Run:

31 42 73 24 15 96 78 44 62 20 39 58 40 60 54 88

Principal Diagonal: 31 96 39 88

Secondary Diagonal: 24 78 20 40

28. Create a 2D list of size 4 x 3. Means, there are four rows and three columns. Initialize elements at random with two-digit numbers. Print elements in single line. Next, print elements in tabular form. Print sum of each row at the end of each row:

Sample Run:

31 42 73 24 15 96 78 44 62 20 39 58

31 42 73 = 146
24 15 96 = 135
78 44 62 = 184
20 39 58 = 117

(Practice 28)

29. Create a list/ array of 10 elements with random values of your choice. Print values in reverse order that is print array/ list from last element to first element
30. Write a program to find the sum of all elements of the array
31. Write a program to copy array/ list into another array/ list
32. Take two arrays of five elements each, where elements are in ascending order. Create another array of 10 elements. Write code to merge previous two arrays into new array in ascending order
33. Write a program to declare an array of ten elements. Initialize them randomly in any range. Print even and odd elements in separate rows. Count both elements. If there are more even elements, make all elements even by adding one or if there are more odd elements, make all elements odd by subtracting one. At the end print all elements in a single line.
34. Write a program, declare an array of ten elements. Initialize them randomly with values 1 to 15, where value should not repeat.
35. Write a program, declare an array of ten elements. Initialize them randomly with values 1 to 5. Check and print index of same element exists at different locations.
36. Write a program, declare an array of 20 elements. Initialize elements randomly such that each element should be larger than previous element. Print the missing values in the list. See sample run carefully:

```
1 3 4 6 7 8 9 13 15 18 20 22 24 26 33 35 37 38 42 45
```

Output:

```
2 5 10 11 14 16 18 19 21 23 27 28 29 30 31 32 34 36 39 40 41 43 44
```

37. Write a program, declare an array of 20 elements. Initialize elements at random in range 0-9. Next, average out each element with value of 4 neighbors. Two neighbors from left side and two from right side. Leave first two values and last two values, see explanation to get understanding:

Explanation:

```
 3  2  0  1  2  4  6  2  1  9  8  2  3  4  6  2  0  1  3  4
```

3 2 2 1 3 3 2 3 5 4 3 3 4 3 2 1 1 2 3 4

First line has 20 random values in range 0-9. In second row, first two and last two values are same. The third value is average of first, second, fourth and fifth value. The sum is (3+2+1+2) = 8 / 4 = 2. Consider next value. The sum is (2 + 0 + 1 +2) = 5 / 4 = 1.

Each time we will consider updated values, means we will use same array / list for calculation and modification. Next, sum is (2 + 1 + 4 + 6) = 13 / 4 = 3. Similarly, consider last value. The sum is (1 + 1 + 3 + 4) = 9 / 2 = 2

38. Initialize an array of 10 elements with random values in range 3-7. For each element print stars in single line. Consider sample run for understanding:

Sample Run:

```
3 6 4 1 ...
***
******
****
*
```

39. Initialize an array of 10 elements with random values in range 3-7. Print elements of array in triangular style:

Sample Run:

```
3 6 4 1 0 2 7 1 2 3
3
3 6
3 6 4
3 6 4 1
3 6 4 1 0
...
```

40. Modify task 11, print sum of elements with elements:

Sample Run:

```
3 6 4 1 0 2 7 1 2 3
3 = 3
3 6 = 9
3 6 4 = 13
3 6 4 1 = 14
3 6 4 1 0 = 14
...
```

41. Modify task 11 and print elements in triples. There will be 8 triples for ten elements:

Sample Run:

```
3 6 4 1 0 2 7 1 2 3
3 6 4
6 4 1
4 1 0
1 0 2
```

. . .

42. Create a 2D list of size 10 x 10. Initialize value at random in range 0 - 9. Print values in tabular form. Next, print principal diagonals only in diagonal form:

Sample Run:

```
 8  0  6  5  5  7  4  2  0  5
 4  2  9  3  1  1  9  6  1  7
 6  1  8  8  9  0  0  0  2  2
 5  3  6  9  8  4  9  1  7  9
 7  1  2  1  8  3  8  0  7  6
 8  2  5  6  8  8  2  3  7  6
 2  3  7  3  9  3  7  3  9  2
 0  6  0  2  1  0  0  2  5  9
 0  4  6  1  7  2  1  9  7  2
 8  4  9  6  9  9  8  2  8  4


 8
    2
       8
          9
             8
                8
                   7
                      2
                         7
                            4
```

43. Modify task 13. Run nested loop and replace zeros with ones and print values again:

Sample Run:

```
 8  0  6  5  5  7  4  2  0  5
 4  2  9  3  1  1  9  6  1  7
 6  1  8  8  9  0  0  0  2  2
 5  3  6  9  8  4  9  1  7  9
 7  1  2  1  8  3  8  0  7  6
 8  2  5  6  8  8  2  3  7  6
 2  3  7  3  9  3  7  3  9  2
 0  6  0  2  1  0  0  2  5  9
 0  4  6  1  7  2  1  9  7  2
 8  4  9  6  9  9  8  2  8  4

 8  1  6  5  5  7  4  2  1  5
 4  2  9  3  1  1  9  6  1  7
 6  1  8  8  9  1  1  1  2  2
 5  3  6  9  8  4  9  1  7  9
 7  1  2  1  8  3  8  1  7  6
 8  2  5  6  8  8  2  3  7  6
 2  3  7  3  9  3  7  3  9  2
 1  6  1  2  1  1  1  2  5  9
 1  4  6  1  7  2  1  9  7  2
 8  4  9  6  9  9  8  2  8  4
```

44. Modify task 13 (not 14). Print indexes of zero values:

Sample Run:

```
 8  0  6  5  5  7  4  2  0  5
 4  2  9  3  1  1  9  6  1  7
 6  1  8  8  9  0  0  0  2  2
 5  3  6  9  8  4  9  1  7  9
 7  1  2  1  8  3  8  0  7  6
 8  2  5  6  8  8  2  3  7  6
 2  3  7  3  9  3  7  3  9  2
 0  6  0  2  1  0  0  2  5  9
 0  4  6  1  7  2  1  9  7  2
 8  4  9  6  9  9  8  2  8  4

0 1
0 8
2 5
2 6
2 7
4 7
...
```

(Practice 29)

Note: Use file "useme.c" for the tasks:

45. Count the number of words in the paragraph. Use space and null character for counting. Ignore following characters:

    - comma
    - opening & closing parenthesis
    - full stop

46. Extend previous task, find length of maximum length of the word by counting length of each word, and again ignore the characters (mentioned in previous task)

47. Count the number of sentences by using full stop and null character. Print each sentence in separate line.

Next tasks are independent of previous tasks:

48. Take input in two different strings. Concatenate first and second string in third string . Print third string to show proper concatenation.

49. Take input your complete name (with space) in a string. Separate first word of your name in first string and second word of your name in second string. Print both first and second string in separate lines.

(Practice 30)

50. Consider a 2D list/ array of size 9x9, initialized with ones and zeros. Ones indicates path and zero indicates a hurdle. It is given that first value at 0, 0 is always 1. Write a function print path from 0, 0 to 8, 8. If there exist a path, otherwise print the maximum possible path.

See example:

```
1  0  0  1  0  1  1  0  0
1  0  0  0  0  1  0  0  0
1  0  0  0  1  1  1  1  1
1  0  0  0  0  1  0  0  0
1  0  0  0  1  0  1  0  0
```

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	1	1	0	0	1	1	1	**1**
0	0	1	1	1	1	1	1	**1**
0	1	1	1	0	1	1	1	**1**

Here, you can see a clear path exists from 0, 0 to 8, 8.

1	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	1	0	1
1	0	1	0	0	0	1	1	1
1	0	1	0	0	0	1	1	0
1	**1**	0	1	0	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1
1	1	0	0	1	1	0	1	1

In this example, there is path from 0, 0 to 5, 1 only. Afterwards there is no adjacent one to move further.

You can move to adjacent right or down, if there is one in the adjacent cell. If both right and down adjacent cells have one, the preference is to move to right. If none of them has one, then stop.

Sample Runs:

1 1 0 0 1 0 0 0 0
0 1 1 1 1 1 1 1 0
1 0 1 0 0 1 0 0 1
0 0 1 1 0 1 0 1 0
1 1 1 0 1 1 0 1 1
0 1 1 1 0 1 0 0 1
1 1 1 1 1 0 0 1 0
0 0 0 0 0 0 0 0 1
0 1 0 0 0 1 1 0 1
0, 0 - 0, 1 - 1, 1 - 1, 2 - 1, 3 - 1, 4 - 1, 5 - 1, 6 - 1, 7 - path blocked

---

1 0 0 0 1 1 0 0 1
0 0 0 0 1 0 1 0 0
0 1 1 0 0 0 1 1 1
1 0 1 1 0 1 1 0 0
1 1 1 1 1 1 0 1 1
1 0 1 0 1 1 1 1 1
0 0 0 1 1 0 1 0 0
1 1 1 0 1 1 1 1 1
1 0 1 1 0 0 1 1 0
0, 0 - path blocked

---

1 0 1 0 1 1 1 1 1
1 1 0 0 0 0 1 1 0
1 1 1 0 1 1 1 1 0
0 0 1 0 0 0 0 0 1
1 0 1 1 0 1 1 1 0
0 0 0 1 1 0 1 1 0
1 0 1 0 1 1 0 0 1
1 0 0 0 0 1 1 0 0

1 1 0 0 0 0 1 1 1
0, 0 - 1, 0 - 1, 1 - 2, 1 - 2, 2 - 3, 2 - 4, 2 - 4, 3 - 5, 3 - 5, 4 - 6, 4 - 6, 5 - 7, 5 - 7, 6 - 8, 6 - 8, 7 - 8, 8 -

51. Consider a 2D list/ array of size 9x9. Ignore boundary values that is first and last row & first and last column. Write function to average out remaining values by putting average of eight neighbors.

1 2 8 5 4 7 6 1 6
9 **1** 4 1 4 7 8 8 8
4 5 4 6 8 8 5 6 9
6 7 3 2 4 9 4 2 9
8 1 2 2 7 9 4 9 3
5 6 1 8 1 1 5 8 6
1 6 6 2 6 1 5 1 2
8 6 2 2 1 4 1 9 3
8 2 5 5 5 7 3 8 1

[For element on second row and second column that is 1, the eight neighbors are, 1, 2, & 8 in first row. 9 & 4 in second row and 4, 5 & 4 in third row. Their sum 1 + 2 + 8 + 9 + 4 + 4 + 5 + 4 = 37. 37 / 8 = 4. Therefore, you can see the corresponding value below the line is 4.

---

1 2 8 5 4 7 6 1 6
9 **4** 4 5 6 6 6 6 8
4 5 4 4 5 6 5 5 9
6 4 3 3 5 6 6 5 9
8 4 4 3 4 4 5 6 3
5 5 4 4 3 3 3 4 6
1 4 4 2 3 2 4 4 2
8 4 3 3 3 4 4 4 3
8 2 5 5 5 7 3 8 1

52. Create a Tic-Tac-Toe two player game. You may take a 2D list/ array. You may store '-' in all nine positions. Later take input from players one-by-one. The player has to select a valid i, j position (which is empty and within range). However, the player may give wrong input, so you have to check. If input is valid, you may store 'A' for first player and 'B' for second player in corresponding cell. After each change, redraw the board to show updated status of the game.

After each turn, check the win status, it is possible that a player succeed to occupy any row, column or diagonal on the board. If a player becomes successful to occupy a winning position, print the player A win or player B win and stop the game. Otherwise, after every cell is filled and no player wins, stop game and print draw game message. See sample runs for better understanding:

Sample Runs:

```
- - -
- - -
- - -
Enter input row and column separated by space:0 0
a - -
- - -
- - -
Enter input row and column separated by space:1 1
a - -
- b -
- - -
Enter input row and column separated by space:2 0
a - -
- b -
a - -
Enter input row and column separated by space:1 2
```

```
a - -
- b b
a - -
Enter input row and column separated by space:1 0
```
Congratulations, Play A won the game
```
                        ------------------------------------
- - -
- - -
- - -
Enter input row and column separated by space:0 0
a - -
- - -
- - -
Enter input row and column separated by space:0 1
a b -
- - -
- - -
Enter input row and column separated by space:0 2
a b a
- - -
- - -
Enter input row and column separated by space:1 0
a b a
b - -
- - -
Enter input row and column separated by space:1 1
a b a
b a -
- - -
Enter input row and column separated by space:1 2
a b a
b a b
- - -
Enter input row and column separated by space:2 1
a b a
b a b
- a -
Enter input row and column separated by space:2 0
a b a
b a b
b a -
Enter input row and column separated by space:2 2
```
Congratulations, Play A won the game
```
                        ------------------------------------
- - -
- - -
- - -
Enter input row and column separated by space:1 1
- - -
- a -
- - -
Enter input row and column separated by space:0 0
b - -
- a -
- - -
Enter input row and column separated by space:2 2
b - -
```

```
- a -
- - a
Enter input row and column separated by space:0 1
b b -
- a -
- - a
Enter input row and column separated by space:2 1
b b -
- a -
- a a
Enter input row and column separated by space:0 2
```
Congratulations, Play B won the game

```
- - -
- - -
- - -
Enter input row and column separated by space:0 0
a - -
- - -
- - -
Enter input row and column separated by space:1 1
a - -
- b -
- - -
Enter input row and column separated by space:0 1
a a -
- b -
- - -
Enter input row and column separated by space:0 2
a a b
- b -
- - -
Enter input row and column separated by space:1 0
a a b
a b -
- - -
Enter input row and column separated by space:2 0
```
Congratulations, Play B won the game

```
- - -
- - -
- - -
Enter input row and column separated by space:0 0
a - -
- - -
- - -
Enter input row and column separated by space:1 1
a - -
- b -
- - -
Enter input row and column separated by space:2 2
a - -
- b -
- - a
Enter input row and column separated by space:0 2
a - b
- b -
```

```
- - a
Enter input row and column separated by space:2 0
a - b
- b -
a - a
Enter input row and column separated by space:1 0
a - b
b b -
a - a
Enter input row and column separated by space:1 2
a - b
b b a
a - a
Enter input row and column separated by space:2 1
a - b
b b a
a b a
Enter input row and column separated by space:0 1
```

Game Draw

(Practice 31) Practice 31 - File Handling - 2D List/ Array

The file has number of values in first line.
The next two lines have two set of values.

53. **Task 01:** Read values in two lists. Create a new file "sum.txt", write number of values in the first line of new file. Sum values of two lists and write sum into a single line of new file. The resultant new file should be like:

20
3 7 3 4 4 5 3 5 5 5 2 4 8 7 2 2 7 8 5 7 5 5 7 7 3

54. **Task 02:** Again, create a new file "compare.txt". Store number of values in first line. Compare corresponding elements in two lists. For each comparison store 0 in the file, if elements are same, otherwise store 1. Write result of all comparisons in single line. The resultant new file should be like:

20
0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0

55. **Task 03:** Create a new file ""compare1.txt". Store number of values in first line. Compare corresponding elements in two lists. For each comparison store >, if first list has larger value, store <, if first list has smaller value, otherwise store =. Write result of all comparisons in single line. The resultant new file should be like:

20
> < < = < < < < > < = = = > = = > = < < > < < > >

Use map.txt to do next tasks, read file and create 2D list using following code:

```
def main():
      file = open('map.txt','r')
      SIZE = int(file.readline())
      map =[[0 for i in range(SIZE)] for j in range(SIZE)]
      for i in range(SIZE):
            s = file.readline().split()
            for j in range(len(s)):
                  map[i][j] = int(s[j])
      file.close()
      print(map)
```

```
9
0 3 3 1 5 4 1 5 1
```

```
5 0 2 1 1 1 5 3 -1
4 2 0 4 2 3 4 3 -1
4 3 3 0 4 2 1 3 4
-1 2 4 1 0 1 -1 4 1
2 -1 -1 4 2 0 5 2 3
-1 4 1 -1 -1 2 0 2 -1
2 5 3 3 -1 1 2 0 2
4 -1 4 4 1 1 -1 -1 0
```

You may cross check output of next tasks from above values (These values are from map.txt). In next tasks the output is actual output not sample run.

56. Assume there are 9 cities and map has path from each city to every other city, where -1 shows there is no direct path. For each City count and print number of direct paths.

```
City 0 has 8 direct path(s) to other Cities
City 1 has 7 direct path(s) to other Cities
City 2 has 7 direct path(s) to other Cities
City 3 has 8 direct path(s) to other Cities
City 4 has 6 direct path(s) to other Cities
City 5 has 6 direct path(s) to other Cities
City 6 has 4 direct path(s) to other Cities
City 7 has 7 direct path(s) to other Cities
City 8 has 5 direct path(s) to other Cities
```

57. **Task 05:** Count and print Cities having direct path coming from other cities. For example, For City 0 there are 6 direct paths coming.

```
City 0 has 6 direct path(s) from other Cities
City 1 has 6 direct path(s) from other Cities
City 2 has 7 direct path(s) from other Cities
City 3 has 7 direct path(s) from other Cities
City 4 has 6 direct path(s) from other Cities
City 5 has 8 direct path(s) from other Cities
City 6 has 6 direct path(s) from other Cities
City 7 has 7 direct path(s) from other Cities
City 8 has 5 direct path(s) from other Cities
```

58. **Task 06:** For each city compare distance with other cities and print the distance and city number, where cities have same distance with source city. See output carefully:

```
City 0 has distance 3 with City: 1 2
City 0 has distance 1 with City: 3 6 8
City 0 has distance 5 with City: 4 7
City 0 has distance 4 with City: 5
City 1 has distance 5 with City: 0 6
City 1 has distance 2 with City: 2
City 1 has distance 1 with City: 3 4 5
City 1 has distance 3 with City: 7
City 2 has distance 4 with City: 0 3 6
City 2 has distance 2 with City: 1 4
City 2 has distance 3 with City: 5 7
City 3 has distance 4 with City: 0 4 8
City 3 has distance 3 with City: 1 2 7
City 3 has distance 2 with City: 5
City 3 has distance 1 with City: 6
City 4 has distance 2 with City: 1
City 4 has distance 4 with City: 2 7
City 4 has distance 1 with City: 3 5 8
City 5 has distance 2 with City: 0 4 7
City 5 has distance 4 with City: 3
City 5 has distance 5 with City: 6
```

```
City 5 has distance 3 with City: 8
City 6 has distance 4 with City: 1
City 6 has distance 1 with City: 2
City 6 has distance 2 with City: 5 7
City 7 has distance 2 with City: 0 6 8
City 7 has distance 5 with City: 1
City 7 has distance 3 with City: 2 3
City 7 has distance 1 with City: 5
City 8 has distance 4 with City: 0 2 3
City 8 has distance 1 with City: 4 5
```