---

1. For given 2D array, write a function that accepts two parameters I & J, where I and J represent rows & columns in the array (e.g., when I = 0 and J = 0, we are referring to cell at first row & first column). For any given cell (represented by I & J), the function must check how many times the value contained in this cell has occurred in that particular row. Pass initial values of I and J in the function from main function.

   Example:
   Given array:

12	31	21	11	5
11	97	14	31	2
91	57	46	57	46
12	8	10	25	83
2	38	61	39	40

   Passing 2 and 3 from main for values of I and J

   Output:
   No. of occurrences of *57* found: **2**

2. Write a program that has an integer array in the main function. Take size of the array as input from the user to create it. Your program should also take some integer values in the range 25 to 75 as input from the user to store in the array. To store these values, pass them sequentially to a function which takes an integer parameter and returns an index of the array based on the following formula:

   index = (value + 3 * value) % array size

   The integer values can then be sequentially saved to the array according to the corresponding index returned for them. However, while storing each value, the following conditions must be taken into account:

   i. If there is already a value stored at that index, this situation is called 'clash'. In this case, try to store the value at the next index. If it's also occupied, keep checking until you find an available slot
   ii. The array can be treated as being circular in nature i.e., if last index is occupied, then index 0 can be checked and it its available, it can be used to store the value.

Your program should display what is stored at each index of the array (including empty cells). It should also display how many times a 'clash' occurred for the given input values.

3. Write a program that contains a float array of dimensions 10x10. Initialize the values of this array in a loop by user-provided values. Then find (and print) the average value of each row and each column.

4. Write a program to implement Tic-Tac-Toe game. You can consider the following suggestions while designing your program:

    i. Use a 2D character array of 3x3 dimensions.
    ii. Use characters ‘x’ and ‘o’ as markers and treat any other character as invalid input.
    iii. Use a loop where each iteration could represent a player’s turn.
    iv. Use conditions to check if the position in the grid selected by a player is already marked, and also conditions for ensuring that invalid position is not selected.

5. Create a size-20 1D array of data type int or float and use a loop to initialize it. Your task is to sort the first 10 elements of this array in ascending order and the last 10 elements in descending order.