Lab-11

Instructions:

- Indent your code properly.
- Use meaningful variable names. Follow the naming conventions.
- Use meaningful prompt lines/labels for all input/output that is done by your programs.
- You are not allowed to discuss your problems with your fellows

Task-01

Write a C++ function which has the following prototype:

int compareAndCount (int a1[], int a2[], int n);

In the above prototype, **a1** is the first array, **a2** is the second array, and **n** is the size of both of these arrays (assume that both arrays have the same number of elements). This function should compare the elements of two arrays (index-by-index) and determine (and return) the count of the elements that are same in both of these arrays. For example, if **a1** contains **{ 4, 7, 3, 8, 1 }** and **a2** contains **{ 2, 7, 6, 8, 4 }**, then the above function should return 2, because 2 elements (**7** which is at index 1, and **8** which is at index 3) are same in these two arrays.

Task 02:

Write a program that contains following functions:

1- **bool addArrays(int arr1[], int s1, int arr2[], int s2)** it takes two arrays and their sizes and if the sizes of both arrays are same then adds arr2 into arr1 and returns true otherwise it does not perform addition and returns false.

2- **bool addArrays(int arr1[][..], int r1, int c1, int arr2[][..], int r2, int c2)** it takes two matrices and their dimensions(i.e. number of rows and columns) and if the dimensions of both 2D arrays are same then adds arr2 into arr1 and returns true otherwise it does not perform addition and returns false.

3- **int main()** it contains required variables and calls the above two functions and displays proper massages.

Task-03

Write a function that will take an array of Characters as parameters and returns true if it is a Palindrome and returns false if it is not.

Sample Input:

Enter a Character Array : abba

Sample output:

It is a Palindrome!

Sample Input:

Enter a Character Array :12345678

Sample output:

It is NOT a Palindrome!

Task-04:

Write a program that creates a two-dimensional array initialized with test data. Use any data type you wish. The program should have the following functions:

- **getTotal**. This function should accept a two-dimensional array as its argument and return the total of all the values in the array.
- **getAverage**. This function should accept a two-dimensional array as its argument and return the average of all the values in the array.
- **getRowTotal**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the total of the values in the specified row.
- **getColumnTotal.** This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The function should return the total of the values in the specified column.
- **getHighestInRow**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the highest value in the specified row of the array.
- **getLowestInRow.** This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the lowest value in the specified row of the array.

Demonstrate each of the functions in this program.

Task-05:

Write a program in which you should:

1- Declare and initialize an integer j and display its address and data on screen.
2- Declare a pointer to integer p and assign the address of j to p. Then display the address and contents of p, compare the contents of p with the address of j.
3- Increment the value of j by using dereference operator with the pointer p.
4- Repeat the above three steps for char and double data types.

Task-06

Write a program in which you should:

1- Declare and initialize an integer array arr[5] and display the address of its first index.
2- Declare a pointer to integer p and assign the contents of arr to p. Take input from user for all of the elements of the array by using pointer p and dereference operator.
3- Display the address and contents of all the elements of the array by using pointer p and dereference operator.
4- Repeat the above three steps for char and double data types.

Task-07

This is to deal with matrices. Write a menu program to perform each operation and make functions separately for each one of the tasks below:

1. Take input in a 2D matrix
2. Display a 2D matrix
3. Perform row major traversal of a 2D matrix and display that.
4. Take transpose of a 2D matrix
5. Take two 2D matrices and add them
6. Take two 2D matrices and multiply them

Instructions for implementation:

- Declare matrices of const size and use that size whenever needed. Your program should be flexible enough that if at the time of testing, we change that const size values, program should work. (For now, use the const size but after reading chapter 9, you need to modify the code for dynamically allocated 2d arrays.)

- Declare the 2d matrices in main(), pass them to functions, perform operation in function and don't use unnecessary cout statements in functions.

- Don't allocate any extra arrays in functions for example in transpose function, save the transpose of the matrix in the same matrix, for addition and multiplication store the result in any of the two passed matices.

Remember: Honesty always gives fruit (no matter how frightening is the consequence); and
Dishonesty is always harmful (no matter how helping it may seem in a ce rtain situation)!