**National University of Computer & Emerging Sciences (NUCES) Islamabad,**

Department of Computer Science

**Programming Fundamentals – Fall 2022**

**LAB 14**

A picture containing text, gear, metalware

Description automatically generated

**Learning Outcomes**

In this lab you are expected to learn the following:

* 2D arrays
* passing 2D arrays to functions
* File handling

**ARRAYS-II(Multi-dimensional and Char arrays)**

In C++, we can create an array of an array, known as a multidimensional array.

For example: **int x[3][4];**

Here, x is a two-dimensional array. It can hold a maximum of 12 elements. We can think of this array as a table with 3 rows and each row has 4 columns as shown below.

Table

Description automatically generated

**Initialization of two-dimensional array**

**int test[2][3] = {2, 4, 5, 9, 0, 19};**

The above method is not preferred. A better way to initialize this array with the same array elements is given below:

**int test[2][3] = { {2, 4, 5}, {9, 0, 19}};**

This array has 2 rows and 3 columns, which is why we have two rows of elements with 3 elements each.

Calendar

Description automatically generated



**Example: Two-Dimensional Array**

// C++ Program to display all elements

// of an initialized two-dimensional array

#include <iostream>

using namespace std;

int main() {

int test[3][2] = {{2, -5}, {4, 0}, {9, 1}};

// use of nested for loop

// access rows of the array

for (int i = 0; i < 3; ++i) {

// access columns of the array

for (int j = 0; j < 2; ++j) {

cout << "test[" << i << "][" << j << "] = " << test[i][j] << endl; }

}

return 0;

}

**Output**

test[0][0] = 2

test[0][1] = -5

test[1][0] = 4

test[1][1] = 0

test[2][0] = 9

test[2][1] = 1

**2D array as Function arguments:**

•Use array name as argument in function call:

**getExams(exams, 2);**

•Use empty [] for row, size declarator for column in prototype, header:

**const int COLS = 2;**

// Prototype

**void getExams(int [][COLS], int);**

// Header

**void getExams(int exams[][COLS], int rows)**

**Example:**

Graphical user interface, text, application, Word

Description automatically generated

**File Handling**

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

In C++ we have a set of file handling methods. These include ifstream, ofstream, and fstream. These classes are derived from fstrembase and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream.

ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files

ifstream: This Stream class signifies the input file stream and is applied for reading information from files

fstream: This Stream class can be used for both read and write from/to files.

All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files.

C++ provides us with the following operations in File Handling:

* Creating a file: open()
* Reading data: read()
* Writing new data: write()
* Closing a file: close()

**Opening a File**

Generally, the first operation performed on an object of one of these classes is to associate it to a real file. This procedure is known to open a file.

We can open a file using any one of the following methods:

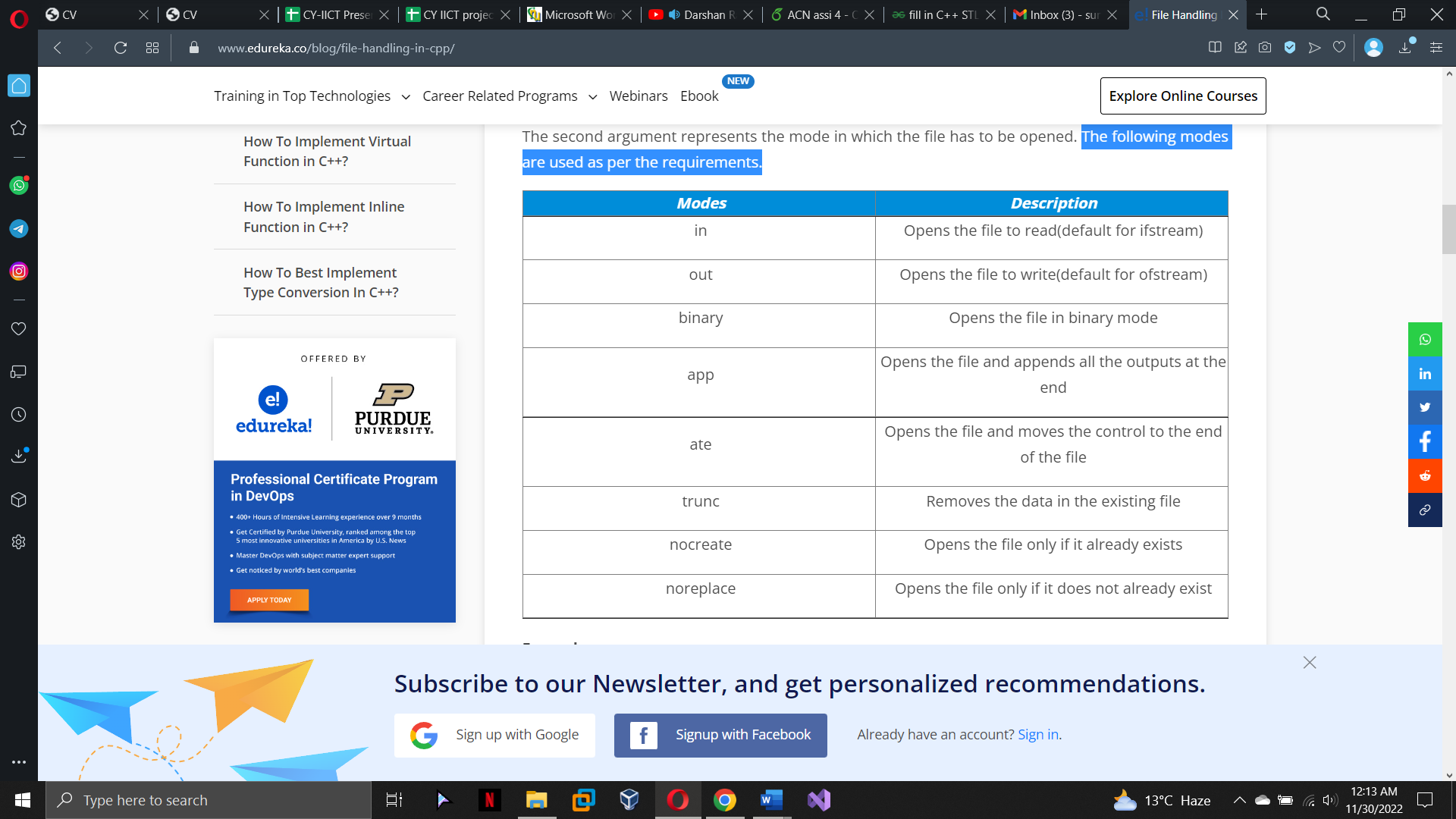
1. First is bypassing the file name in constructor at the time of object creation.

2. Second is using the open() function.

To open a file use: void open(const char\* file\_name,ios::openmode mode);

Here, the first argument of the open function defines the name and format of the file with the address of the file.

The second argument represents the mode in which the file has to be opened. The following modes are used as per the requirements.



**Example of creating a file:**

#include <iostream>

#include<fstream>

using namespace std;

int main()

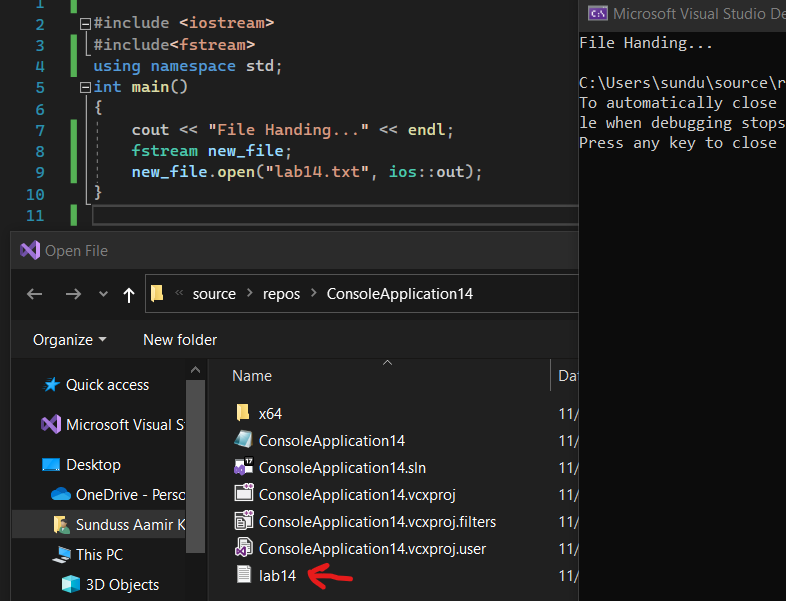
{

cout << "File Handing..." << endl;

fstream new\_file;

new\_file.open("lab14.txt", ios::out);

}



In the above example, new\_file is an object of type fstream, as we know fstream is a class so we need to create an object of this class to use its member functions. So we create new\_file object and call open() function. Here we use out mode that allows us to open the file to write in it.

Default Open Modes :

* ifstream ios::in
* ofstream ios::out
* fstream ios::in | ios::out

We can combine the different modes using or symbol ‘|’.

**Example of opening or creation of file (checking):**

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

fstream new\_file;

new\_file.open("Lab\_14\_Test.txt", ios::out);

if (!new\_file)

{

cout << "File creation failed";

}

else

{

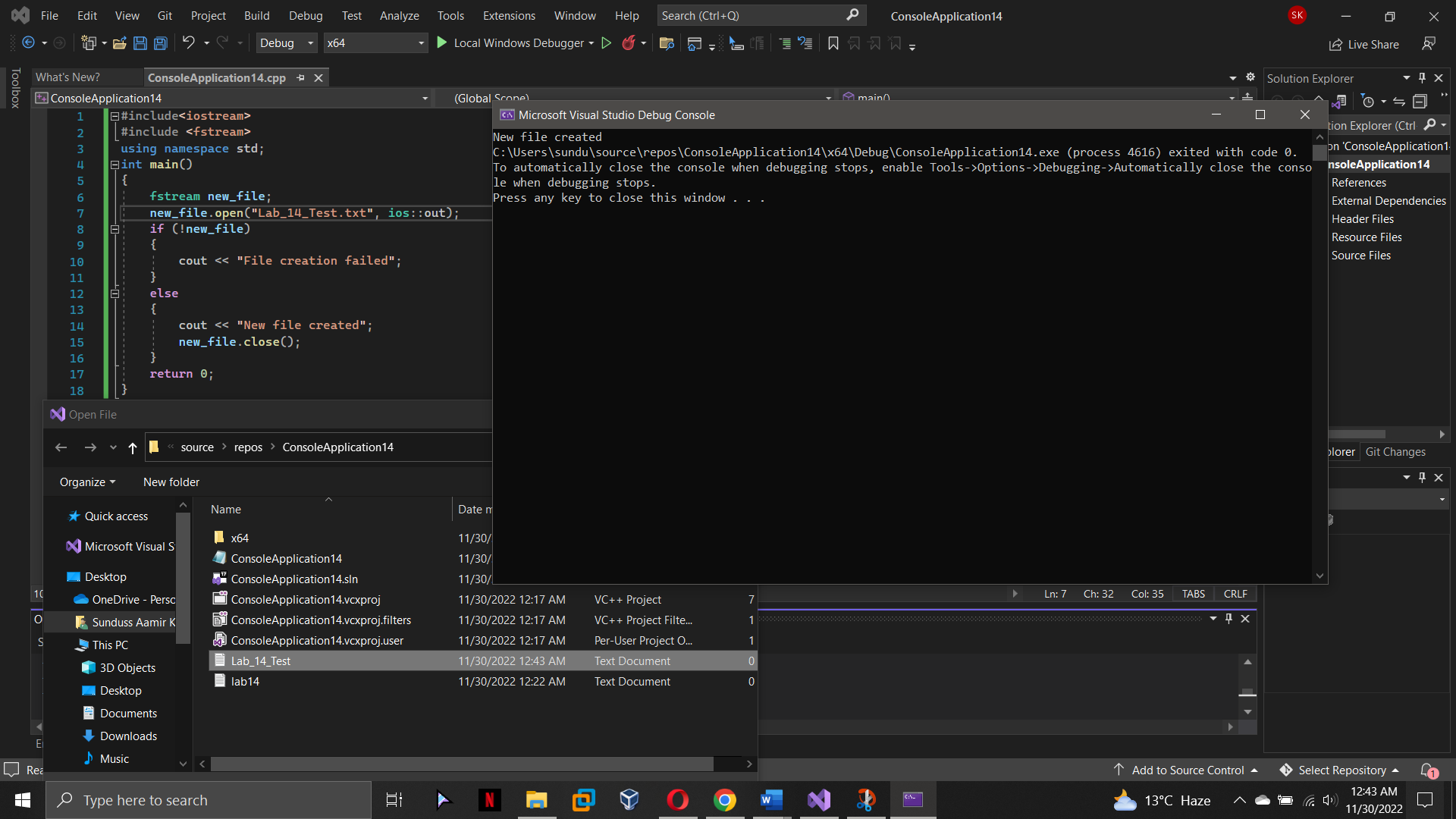
cout << "New file created";

new\_file.close();

}

return 0;

}



In the above example we first create an object to class fstream and name it ‘new\_file’. Then we apply the open() function on our ‘new\_file’ object. We give the name ‘Lab\_14\_Test’’ to the new file we wish to create and we set the mode to ‘out’ which allows us to write in our file. We use a ‘if’ statement to find if the file already exists or not if it does exist then it will going to print “File creation failed” or it will gonna create a new file and print “New file created”.

**Writing in a File:**

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

fstream new\_file;

new\_file.open("lab14.txt", ios::out);

if (!new\_file)

{

cout << "File creation failed";

}

else

{

cout << "New file created";

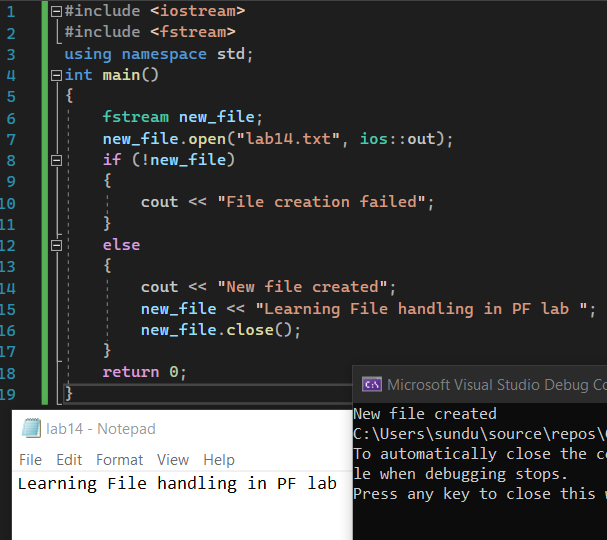
new\_file << "Learning File handling in PF lab ";

new\_file.close();

}

return 0;

}

****

Here we first create a new file “lab14” using open() function since we wanted to send output to the file so, we use ios::out. As given in the program, information typed inside the quotes after Insertion Pointer “<<” got passed to the output file.

**Reading from a File (without spaces):**

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

fstream new\_file;

new\_file.open("lab14.txt", ios::in);

if (!new\_file)

cout << "No such file";

else

{

char ch;

while (!new\_file.eof())

{

new\_file >> ch;

cout << ch;

}

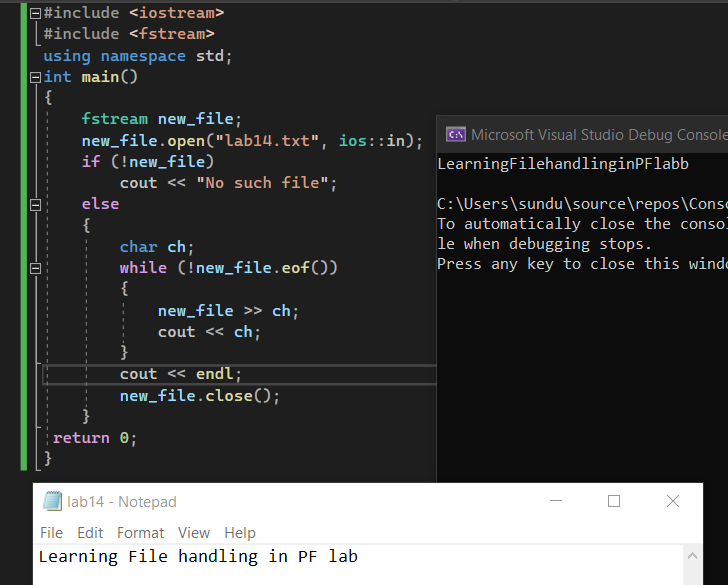
cout << endl;

new\_file.close();

}

return 0;

}



In this example, we read the file that generated id previous example i.e. lab14.

To read a file we need to use ‘in’ mode with syntax ios::in. In the above example, we print the content of the file using extraction operator >>. The output prints without any space because we use only one character at a time, we need to use getline() with a character array to print the whole line as it is.

**Reading from a File (with spaces):**

#include <iostream>

#include <fstream>

#include<string>

using namespace std;

int main()

{

fstream new\_file;

new\_file.open("lab14.txt", ios::in);

if (!new\_file)

cout << "No such file";

]0 else

{

string s;

while (!new\_file.eof())

{

getline(new\_file, s);

cout << s;

}

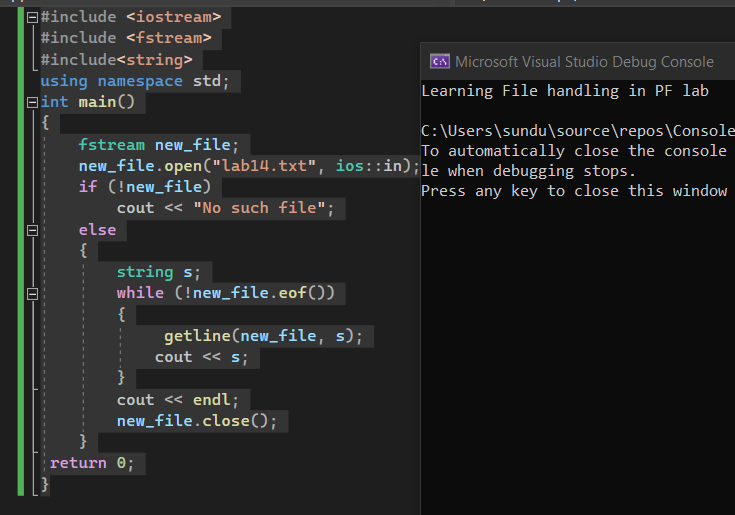
cout << endl;

new\_file.close();

}

return 0;

}

****

**Lab Tasks**

**Task 1:**

Write a program that declares a two-dimensional integer array by the name of matrix with rows=5, columns=5, stores the elements in the array matrix with the help of a nested for loop. Now you have to find individual sum of each row of the 2D array and store corresponding result in 1D array i.e. sum of all the elements of row 0 should be stored in 1st element of 1D array, similarly sum of all elements of the second row of 2D array should be stored at the second index of 1D array. Display the final sum array (i.e., 1D array). Think about the size of 1D array yourself.

**Example:**

**Array:**

2 3 5 3 1

4 5 1 2 1

4 7 3 2 0

2 1 1 5 1

1 7 8 9 0

**Sum array: 14 13 16 10 25**

**Task 2:**

Write a C++ program to take 3x3 integer array as input from the user using nested loops. Now, pass the 2D array to a function named Reverse. In the function, copy all the elements in another array but in reverse order and show the output.

**Task 3:**

Create a program for writing records of 10 students in a text file named as Record.txt. Store Reg no, Full name, Program, and contact of student.

Output should be like this in the file.

Table 1: Student.txt

i90021 Asad Shafique CS 0300124013

i90022 Shahid Afridi SE 0300124090

i90023 Shaheen Shah DS 0300124071

i90024 Babar Azam CY 0300124043

Once all the records are inserted, display all the information onto the console, reading from the Record.txt

**Task 4:**

Write a program that consist of two functions. The Input function takes the input from user about the specific topic entered by the user and at least ten lines about it and store the information into the file named as “info.txt”. The output function counts the number of words in each line and show the count of lines having less than 10 words.

**Task 5:**

Design a program for reading two text, f1.txt and f2.txt and identifying the difference between the lines of both text files, i.e. writing on the screen the lines that differ in both files. You can consider getline function of istream class.

f1.txt f2.txt

This is a test line. This is a test Hello World. Hello Guys.

Output:

Hello World.

Hello Guys

**Bonus Task:**

Write a function named arrayToFile. The function should accept three arguments: the name of a file, a pointer to an int array, and the size of the array. The function should open the specified file in binary mode and write the contents of the array along with its size to the file, and then close the file.

Write another function named fileToArray. This function should accept two arguments: the name of a file, a pointer to an int array. The function should open the specified file in binary mode, read the size of array and should allocate memory to the pointer and then read the contents of the array, and then close the file. This function should return the size of array read from the \_le. Write a complete program that demonstrates these functions by using the arrayToFile function to write an array to the file, and then using the fileToArray function to read the data from the same file After the data is read from the file into the array, display the arrays contents on the screen.