**Problem 4:** Write a function TestMatrixAddition with following prototype:

|  |
| --- |
| Void TestMatrixAddition(); |

Your TestMatrixAddition function should do following:

* Declare three matrices A, B and C and initialize them to zero.
* Input a matrix A from user (using InputMatrix function)
* Input a matrix B from user (using InputMatrix function)
* Call a function AddMatrix (prototype and description given below)
* Output matrix A, B and A+B

Sample Output is given below:

|  |
| --- |
| A =  5 3  8 1  B =  3 4  1 5  A + B =  8 7  9 6 |

AddMatrix takes three matrices A, B and Result as parameters, add the matrices and save the result in Result matrix. Remember two matrices having different sizes cannot be added so you will not call AddMatrix in this situation and print an error message. Prototype of AddMatrix is given below:

|  |
| --- |
| void AddMatrix( int A[][COLS] , int B[][COLS] ,int Result[][COLS] ,int , int ); |

Update your main as code given below and test your MatrixAddition.

|  |
| --- |
| void main()  {  TestMatrixAddition();  } |

*For all the exercises below, you will use your InputArray and PrintArray functions for input and output, write a Test function and call the test function from main as we did in MatrixAddition exercise.*

**Problem 6:** A square matrix in which all the entries below the main diagonal are zero is called upper triangular. Write a function IsMatrixUpperTriangular which takes a matrix and returns true if the matrix is upper triangular and false otherwise.

|  |
| --- |
| bool IsMatrixUpperTriangular( int [][COLS] , int , int ); |

For example matrix A shown below is upper triangular while matrix B is not upper triangular.

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 2 | 7 | 0 |
| 0 | 5 | 0 | -1 |
| 0 | 0 | 7 | 0 |
| 0 | 0 | 0 | 0 |

Matrix A

|  |  |  |  |
| --- | --- | --- | --- |
| -1 | 2 | 7 | 0 |
| 0 | 5 | 0 | -1 |
| -9 | 0 | 7 | 0 |
| 0 | 0 | 0 | 0 |

Matrix B

Help**:** Notice the indices of blue cells which you need to traverse and device the nested loop accordingly.

|  |  |  |  |
| --- | --- | --- | --- |
| (0,0) | 2 | 7 | 0 |
| (1,0) | (1,1) | 0 | -1 |
| (2,0) | (2,1) | (2,2) | 0 |
| (3,0) | (3,1) | (3,2) | (3,3) |

**Problem 8:** Declare a friend list which will be containing names of your friends (2-D array of characters having names of 5 friends where each name can be at max 50 characters long). Following is the syntax:

|  |
| --- |
| const int FRIENDS\_COUNT = 5; // Total number of friends  const int NAME\_LENGHT = 50; // Each name can be 50 characters long at max.  char myFriendList[FRIENDS\_COUNT][NAME\_LENGHT] = {0}; // All the names initializes to null character.  // myFrientList is 5 by 50 array. First dimension of this list is your 5 friends and second dimension is name of each friend which can be 50 characters long at max. |

Now read names of all your friends from console. Following is the syntax

|  |
| --- |
| for(int i=0 ; i<FRIENDS\_COUNT ; i++)  {  cout<<"Friend "<<i+1<<":\t";  cin.getline(myFriendList[i],NAME\_LENGHT); //myFriendList[i] is the name of ith friend and its max length can be 50.  } |

Now display your friend list using following syntax

|  |
| --- |
| for(int i=0 ; i<FRIENDS\_COUNT ; i++)  {  cout<<"Friend "<<i+1<<":\t";  cout<<myFriendList[i]<<endl; //Display ith friends name which is a c-string.  } |

Now you know how we can access name of ith friend (which is a c-string). Using your find string code which we did in CStrings exercises, update your program such that it takes a string from user and finds it in your friend list and displays all the friends who have that string in their name.

//myFriendList[i][j] gives jth character of ith friend’s name

Sample input and output is shown below:

|  |
| --- |
| Friends Name:  Imran Khan  Javed Miandad  Wasim Akram  Aqib Javed  Saqlain Mushtaq  Enter friend which you want to search in list: Javed  Search Result:  Javed Miandad  Aqib Javed |

**Problem 9:** Write a function that takes as parameters a string (char array), a ‘find’ character and a ‘replace’ character. The function should replace all the occurrences of the ‘find’ character with the ‘replace’ character. For example:

string: “I love programming”

find character: ‘m’

replace character: ‘n’

then the string should become “I love progranning”

char arr[25] = { 'a', 'g', 'T', '@', '5', 'h', '%', '1', 'k', 'R', 'B', '7', '&', 'l', 'V', 'j', '#', 'n', 'Q', '3', '!', ')', '}', 'W', 'y' };

Write a program that converts the upper case letters to lower case letters and lower case letters to upper case letters. The numbers should be replaced with N + 5, where N is the number. If this number exceeds single digit, replace it with the unit digit of that number. Other characters should remain as they are. Display the altered array