PF Problems

Q1:

Write a C++ program to modify a 2D array of numbers. For each element in the array, update it by multiplying it with the values of the elements directly above, below, left, and right of it. If an element is on the edge or in a corner and doesn't have neighbors in some directions, use 1 instead for those missing positions.(take the array values from the user).

Q2:
Write output of the following code.

```
#include <iostream>
using namespace std;

int main() {
    int arr[3][3][3] = {
        {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        },
        {
            {10, 11, 12},
            {13, 14, 15},
            {16, 17, 18}
        },
        {
            {19, 20, 21},
            {22, 23, 24},
            {25, 26, 27}
        }
    };

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

                                arr[i][j][k] *= i;
            }
```

```
        }
    }


    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                cout << arr[i][j][k] << " ";
            }
            cout << endl;
        }
        cout << endl;
    }

    return 0;
}
```

TA1
Q1

```
#include <iostream>
#include <cstring>
using namespace std;

void Reverse(char* str) {
    // Step 1: Reverse the entire string
    int length = strlen(str);
    char* start = str;
    char* end = str + length - 1;

    while (start < end) {
        char temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
```

```
    // Step 2: Reverse each word within the reversed string
    start = str;
    while (*start) {
        while (*start == ' ') // Skip leading spaces
            start++;
        char* wordEnd = start;
        while (*wordEnd && *wordEnd != ' ') // Find the end of the word
            wordEnd++;

        // Reverse the word
        end = wordEnd - 1;
        while (start < end) {
            char temp = *start;
            *start = *end;
            *end = temp;
            start++;
            end--;
        }

        start = wordEnd;
    }
}

int main() {
    char str[100];
    cout << "Enter a string: ";
    cin.getline(str, 100);

    cout << "Original Character Array:\n" << str << endl;

    Reverse(str);

    cout << "After calling Function Reverse():\n" << str << endl;

    return 0;
}
```

TA 7:

(easy) Q: Given an NxM array from the user where N is number of rows and M is number of columns, print the transpose of the array.

(easy) Q: Print the 2D array column-wise in a wave pattern (down in the first column, up in the second, and so on). The array would be provided as input by the user.
For e.g for the given 4x4 matrix
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
The output should look like:
1 5 9 13 14 10 6 2 3 7 11 15 16 12 8 4

(med) Q: Given two collections of integers, where both collections are one dimensional arrays given by the user, write a program to compute the union and intersection of the two collections. The union should include all unique integers from both collections, while the intersection should include only the integers that are common to both. (Question is similar to the one from session-II of 23rd batch.

(med/hard) Q: Write a C++ program to determine if it is possible to navigate through a maze from the top-left corner to the bottom-right corner. The maze is represented as a 2D grid of integers, where:

1 represents a path you can traverse.
0 represents a blocked cell you cannot traverse.

You can only move down or right at each step. The program should return "Path is possible" if there exists a valid path from the top-left to the bottom-right of the grid, and "Path is not possible" otherwise. The maze will be provided as input by the user.

For e.g
1 1 0 0
1 0 1 1
0 1 0 0
0 1 1 1

This maze does not have a valid path

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

This maze has a valid path

TA3

```
//////////////////////////////////////////////////
int main()
{
	int arr[4][4] = { 3, 2, 3, 8, 2, 3, 7, 9, 3, 7, 3, 2, 8, 9, 2, 3 };

	int rows = 4, cols = 4;
	bool check = true;
	if (rows != cols)
		check = false;
	for (int i = 0, s = i; i < rows; ++i, s++) {
		if (cout << arr[arr[i][i]][arr[0][0]] <<"\n" && arr[0][0] != arr[i][i])
		{
			check = false;
			break;
		}
		for (int j=i+1; j<cols; ++j, cout<<arr[j-1][i])
		{
			if (arr[i][j] != arr[j][i] && cout<<s--)
			{
				check = false;
				break;
			}
		}
	}

	if (check)
		std::cout << "Abracadabra";
}
```

Output:
3
2383
793
23
Abracadabra

```
//////////////////////////////////////////////////
int main() {
	int nrows = 3, ncols = 4;
	int A[2][3][4] = { {
	{ 1, 3, 2 },
	{ 4, 5 },
	{ 7, 8, 9 } },
	{ { 4 },
	{ 5, 5, 7 },
	{ -2, 3, 4 } }
	};
	int b[2][4] = { { 0 } };
	for (int i = 0; i < 2; ++i) {
		for (int j = 0; j < ncols; ++j)
			for (int k = 0; k < nrows; ++k)
				b[i][j] += A[i][k][j];
	}
	for (int i = 0; i < 2; ++i) {
		for (int j = 0; j < nrows; ++j) {
			for (int k = 0; k < ncols; ++k)
				cout << A[i][j][k] << " ";
			cout << endl;
		}
		for (int j = 0; j < ncols; ++j)
			cout << b[i][j] << " ";
		cout << endl;
	}
	return 0;
}
```

Output:
1 3 2 0
4 5 0 0
7 8 9 0
12 16 11 0
4 0 0 0
5 5 7 0
-2 3 4 0
7 8 11 0

//
```
int main()
{
        int a[3][3] = { { 1,2,3} , { 4,5,6} , {7,8,9} };
        function(a);
        cout << a[2][1];
        return 0;
}
void function(int b[][3])
{
        ++b;
        b[1][1] = 9;
}
```

Output: 9

//
```
int main(){
   int A[2][2] = {{1,3},{2,4}};
   int n = 2;
   int C = 100;
   for(int i=0;i<n;i++)
      for(int j=0;j<n;j++) {
         int Temp = A[i][j] + C;
         A[i][j] = A[j][i];
         A[j][i] = Temp - C;}
      for(int i=0;i<n;i++)
         for(int j=0;j<n;j++)
            cout << A[i][j] << endl;}
```

Output:
1
3
2
4