

Lab manual 9

Submitted to: M. Affan

Name: Muhammad Bin Ahmad

ME-15(A)

Roll No: 480779

## List of all Global Functions

### 1. void table(int number, int n){

```
    if(n>10){  
        return;  
    }  
    cout<<number<<" x "<<n<<" = "<<number*n<<endl;  
    table(number,n+1);
```

```
}
```

### 2. void matrix\_multiplication(int matrix1[3][3],int matrix2[3][3]){

```
    int sum;  
    for(int i=0;i<3;i++){  
        for(int j=0;j<3;j++){  
            sum=0;  
            for(int k=0;k<3;k++){  
                sum=sum+(matrix1[i][k]*matrix2[k][j]);  
            }  
            cout<<sum<<" ";  
        }  
        cout<<endl;  
    }  
}
```

```
}
```

### 3. void array2display(int matrix2[3][3]){

```
    for(int i=0;i<3;i++){  
        for(int j=0;j<3;j++){  
            cout<<matrix2[i][j]<<" ";  
        }  
    }
```

```
        cout<<endl;
    }
}
```

**4. void array1display(int matrix1[3][3]){**

```
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cout<<matrix1[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

**5. void array2 (int matrix2[3][3]){**

```
    cout<<"Enter the elements into the second array.\n";
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cin>>matrix2[i][j];
        }
    }
}
```

**6. void array1 (int matrix1[3][3]){**

```
    cout<<"Enter the elements into the array.\n";
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cin>>matrix1[i][j];
        }
    }
}
```

```

7. void transpose(int matrix1[3][3],int matrix2[3][3]){
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            matrix1[i][j]=matrix2[j][i];
            cout<<matrix1[i][j]<<" ";
        }
        cout<<endl;
    }
}

```

```

8. void matrix_sum(int matrix1[3][3], int matrix2[3][3]){
    for (int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cout<<matrix1[i][j]+matrix2[i][j]<<" ";
        }
        cout<<endl;
    }
}

```

**9.** *int matrix1[3][3]*      (Global variable)

**10.** *int matrix2[3][3]*      (Global variable)

1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.

```
int sum=0;
array1(matrix1);
for (int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        if(i==j){
            sum=sum+matrix1[i][j];
        }
    }
}

for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        cout<<matrix1[i][j]<<" ";
    }
    cout<<endl;
}

cout<<"The sum of the numbers on the left diagonal is "<<sum<<endl;

sum=0;
for (int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        if(abs(i-j)==2 || i+j==2){
            sum+=matrix1[i][j];
        }
    }
}

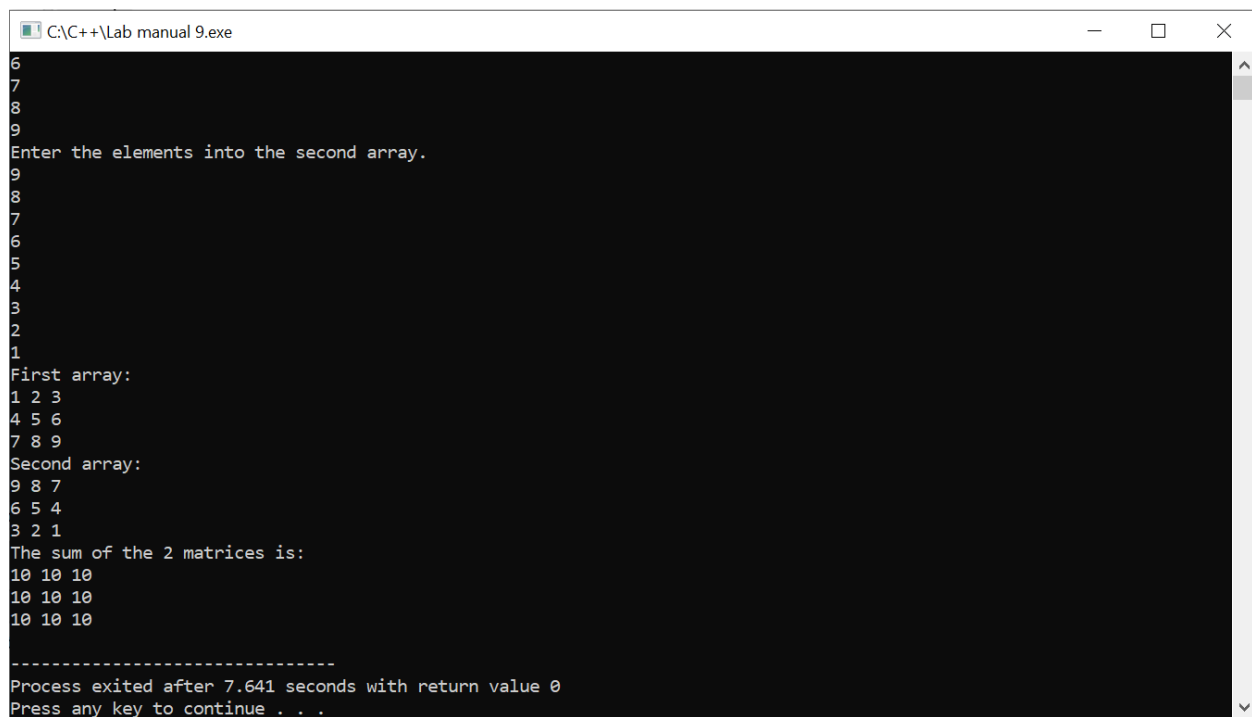
cout<<"The sum of the numbers on the right diagonal is "<<sum<<endl;
```

```
C:\C++\Lab manual 9.exe
Enter the elements into the array.
1
3
5
7
9
7
5
3
1
1 3 5
7 9 7
5 3 1
The sum of the numbers on the left diagonal is 11
The sum of the numbers on the right diagonal is 19

-----
Process exited after 4.136 seconds with return value 0
Press any key to continue . . .
```

2. Write a function to add two 2D arrays of size 3x3.

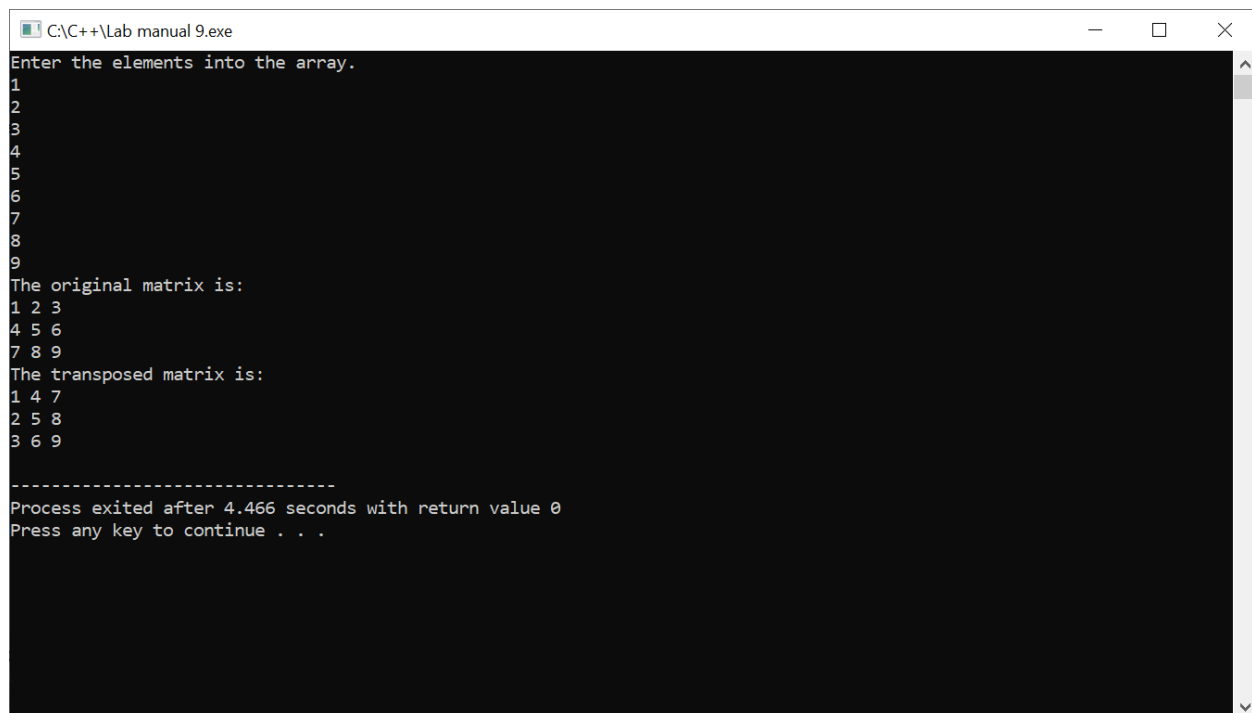
```
array1(matrix1);  
array2(matrix2);  
cout<<"First array:\n";  
array1display(matrix1);  
cout<<"Second array:\n";  
array2display(matrix2);  
cout<<"The sum of the 2 matrices is: "<<endl;  
matrix_sum(matrix1,matrix2);
```



```
C:\C++\Lab manual 9.exe  
6  
7  
8  
9  
Enter the elements into the second array.  
9  
8  
7  
6  
5  
4  
3  
2  
1  
First array:  
1 2 3  
4 5 6  
7 8 9  
Second array:  
9 8 7  
6 5 4  
3 2 1  
The sum of the 2 matrices is:  
10 10 10  
10 10 10  
10 10 10  
-----  
Process exited after 7.641 seconds with return value 0  
Press any key to continue . . .
```

3. Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.

```
int copy_array[3][3];  
array1(matrix1);  
for(int i=0;i<3;i++){  
    for(int j=0;j<3;j++){  
        copy_array[i][j]=matrix1[i][j];  
    }  
}  
  
cout<<"The original matrix is: \n";  
array1display(matrix1);  
cout<<"The transposed matrix is: \n";  
transpose(matrix1,copy_array);
```



```
C:\C++\Lab manual 9.exe  
Enter the elements into the array.  
1  
2  
3  
4  
5  
6  
7  
8  
9  
The original matrix is:  
1 2 3  
4 5 6  
7 8 9  
The transposed matrix is:  
1 4 7  
2 5 8  
3 6 9  
-----  
Process exited after 4.466 seconds with return value 0  
Press any key to continue . . .
```



4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

```
int sum;
```

```
array1(matrix1);
```

```
array2(matrix2);
```

```
cout<<"The first array is:\n";
```

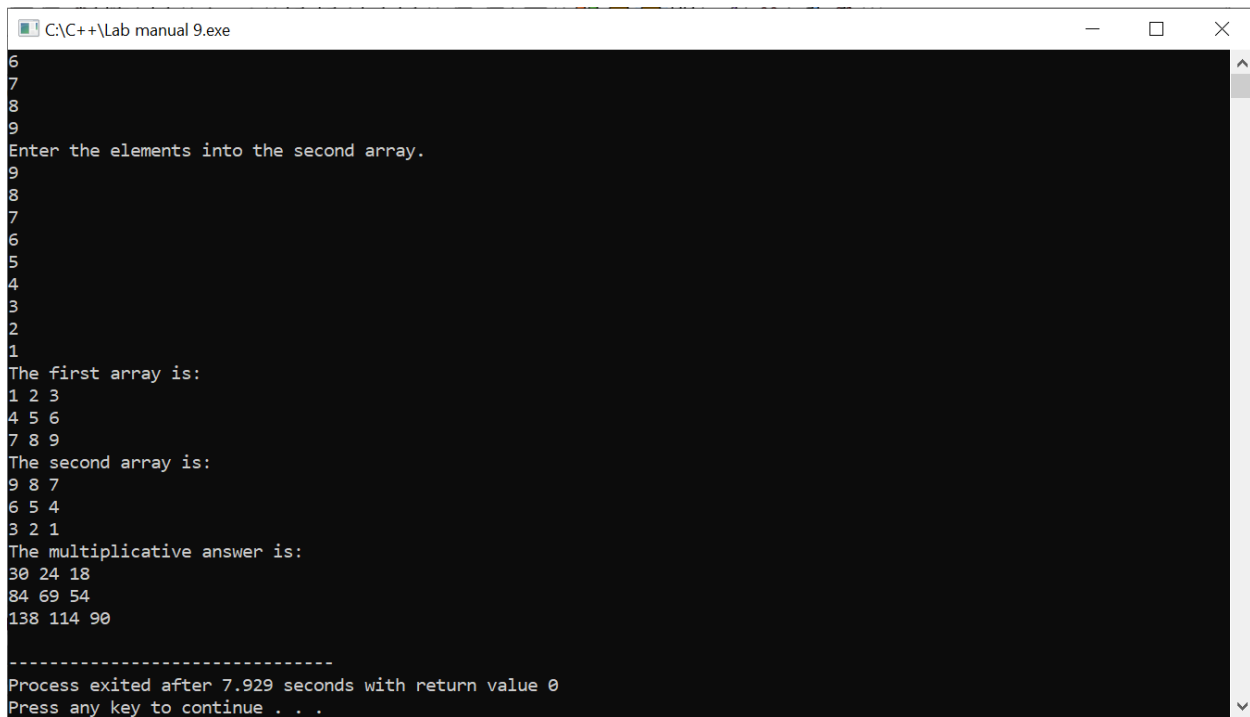
```
array1display(matrix1);
```

```
cout<<"The second array is:\n";
```

```
array2display(matrix2);
```

```
cout<<"The multiplicative answer is: \n";
```

```
matrix_multiplication(matrix1,matrix2);
```



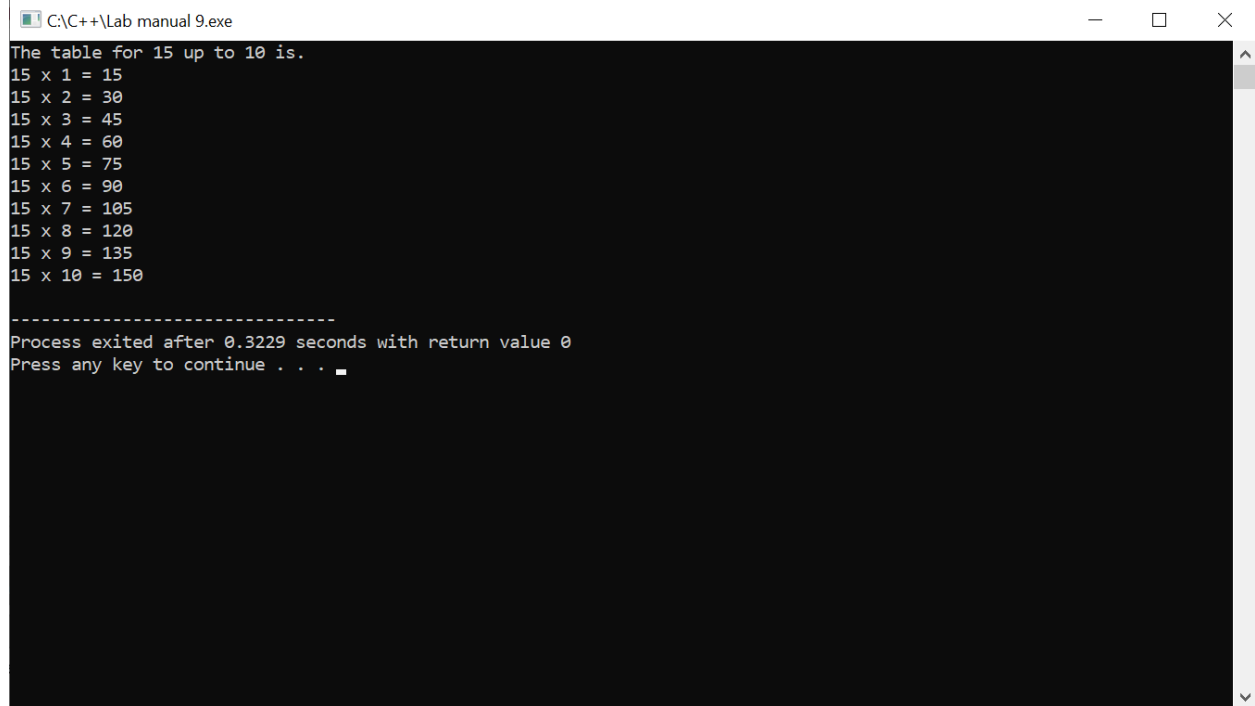
```
C:\C++\Lab manual 9.exe
6
7
8
9
Enter the elements into the second array.
9
8
7
6
5
4
3
2
1
The first array is:
1 2 3
4 5 6
7 8 9
The second array is:
9 8 7
6 5 4
3 2 1
The multiplicative answer is:
30 24 18
84 69 54
138 114 90
-----
Process exited after 7.929 seconds with return value 0
Press any key to continue . . .
```

5. Print the multiplication table of 15 using recursion.

```
int num=15;
```

```
cout<<"The table for 15 up to 10 is.\n";
```

```
table(num,1)
```

A screenshot of a Windows command prompt window titled "C:\C++\Lab manual 9.exe". The window displays the output of a C++ program. The first line is "The table for 15 up to 10 is.". This is followed by ten lines of multiplication results: "15 x 1 = 15", "15 x 2 = 30", "15 x 3 = 45", "15 x 4 = 60", "15 x 5 = 75", "15 x 6 = 90", "15 x 7 = 105", "15 x 8 = 120", "15 x 9 = 135", and "15 x 10 = 150". Below these, a separator line of dashes is shown, followed by the text "Process exited after 0.3229 seconds with return value 0" and "Press any key to continue . . .". The cursor is positioned at the end of the last line.

```
C:\C++\Lab manual 9.exe
The table for 15 up to 10 is.
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150

-----
Process exited after 0.3229 seconds with return value 0
Press any key to continue . . .
```

## HOME TASK

Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.

```
float adjoint[3][3],inverse[3][3];

array1(matrix1);

cout<<"First matrix: \n";

array1display(matrix1);

float determinant =
matrix1[0][0] * (matrix1[1][1] * matrix1[2][2] - matrix1[2][1] * matrix1[1][2]) -
matrix1[0][1] * (matrix1[1][0] * matrix1[2][2] - matrix1[2][0] * matrix1[1][2]) +
matrix1[0][2] * (matrix1[1][0] * matrix1[2][1] - matrix1[2][0] * matrix1[1][1]);

if (determinant == 0) {

cout << "The matrix is singular, it\'s inverse does not exist." << endl;

}

else{

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

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

            adjoint[i][j] = (matrix1[(j+1)%3][(i+1)%3] * matrix1[(j+2)%3][(i+2)%3] -
                matrix1[(j+1)%3][(i+2)%3] * matrix1[(j+2)%3][(i+1)%3]);

        }

    }

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

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

            inverse[i][j] = adjoint[i][j] / determinant;

        }

    }

    cout << "The inverse of the matrix is:" << endl;

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

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

```
C:\C++\Lab manual 9.exe
Enter the elements into the array.
1
3
5
7
9
7
5
3
1
First matrix:
1 3 5
7 9 7
5 3 1
The inverse of the matrix is:
0.25 -0.25 0.5
-0.583333 0.5 -0.583333
0.5 -0.25 0.25
-----
Process exited after 5.331 seconds with return value 0
Press any key to continue . . .
```

```
C:\C++\Lab manual 9.exe
Enter the elements into the array.
1
2
3
4
5
6
7
8
9
First matrix:
1 2 3
4 5 6
7 8 9
The matrix is singular, it's inverse does not exist.
-----
Process exited after 3.727 seconds with return value 0
Press any key to continue . . .
```