**OOP Lab 4**

Make sure to implement the logic with the given sample main function on Eclipse, but use terminal to execute test cases. Test case cpp file is given for every question.

**Task1: Write a C++ program to find the sum of the right diagonal of square matrix.**

```
int main() {
    int n;
    cout << "Enter the size of the square matrix: ";
    cin >> n;

    int **matrix;
    allocateMatrix(matrix,n);
    inputMatrix(matrix,n);
    displayMatrix(matrix,n);
    int sum = sumOfRightDiagonal(matrix,n);
    deallocateMatrix(matrix,n);

    // Display the sum of the right diagonal
    cout << "Sum of the right diagonal is: " << sum << endl;

    return 0;
}
```

**Task 2: Write a C++ program to find if the two matrices are equal or not?**

```
 int main() {
    int n;
    cout << "Enter the size of the square matrix: ";
    cin >> n;

    int **matrix1;
    int **matrix2;
    allocateMatrix(matrix1,n);
    allocateMatrix(matrix2,n);

    inputMatrix(matrix1,n);
    inputMatrix(matrix2,n);

    displayMatrix(matrix1,n);
    displayMatrix(matrix2,n);

    bool flag = isEqual(matrix1,matrix2,n);
```

```
    deallocateMatrix(matrix1,n);
    deallocateMatrix(matrix2,n);


    cout << "Flag =" << flag << endl;
    return 0;
}
```

**Task 3: Write a C++ program to find the product of two matrices, the allocation of the third matrix only if multiplication is possible?**

**Condition:**
**if (c1 != r2) { cout << "Matrix multiplication is not possible!" << endl; return 0; }**

```
int main(){

    int **matrix1;
    int **matrix2;
    int **matrix3;

    int r1=2,c1=2;
    int r2=2,c2=6;

    allocateMatrix(matrix1,r1,c1);
    allocateMatrix(matrix2,r2,c2);

    inputMatrix(matrix1,r1,c1);
    inputMatrix(matrix2,r2,c2);

    displayMatrix(matrix1,r1,c1);
    cout << endl;
    displayMatrix(matrix2,r2,c2);

    multiply(matrix1,matrix2,matrix3,r1,c1,r2,c2);
    cout << endl;
    displayMatrix(matrix3,r1,c2);
    deallocateMatrix(matrix1,r1);
    deallocateMatrix(matrix2,r2);
    deallocateMatrix(matrix3,r1);

    Return 0;
}
```

**Task: 4**

Create two dynamic integer arrays array1 and array2. Write a function to find the intersection of two arrays and return a resultant array(created dynamically). Each element in the resultant array must be unique and you may return the result in any order.

**Task 5:**

Create a 2d array dynamically, populate it and perform following operation:

Write a function to:

1. find sum of all elements of array.
2. Find the largest element in an array.
3. Search for elements in a 2d array.
4. Find a row with maximum sum.
5. Check if the given 2d matrix is identity matrix or not.Accordingly return true or false.