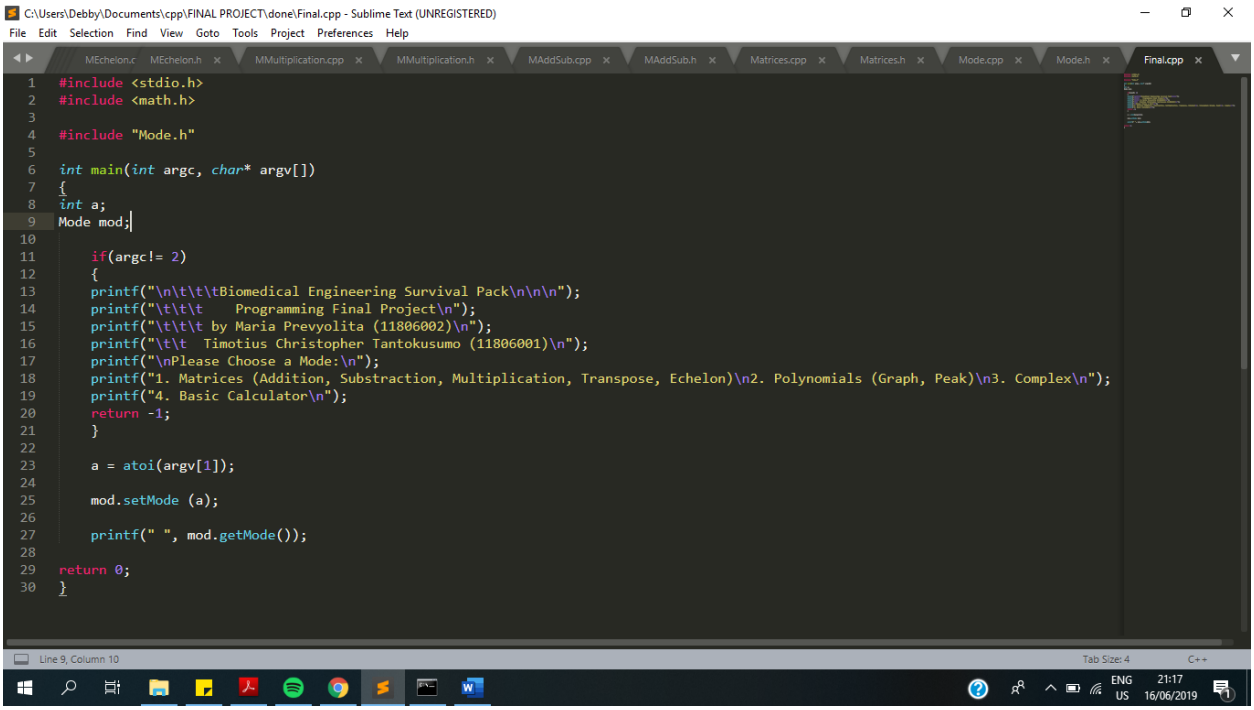
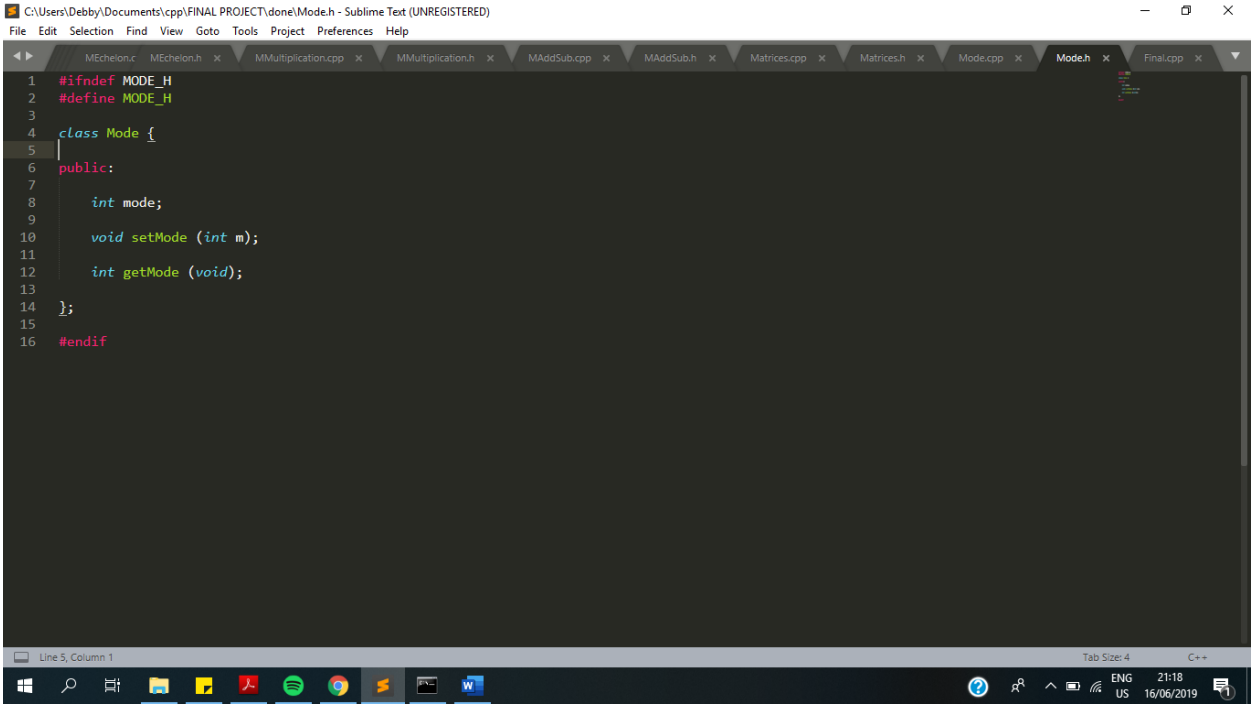
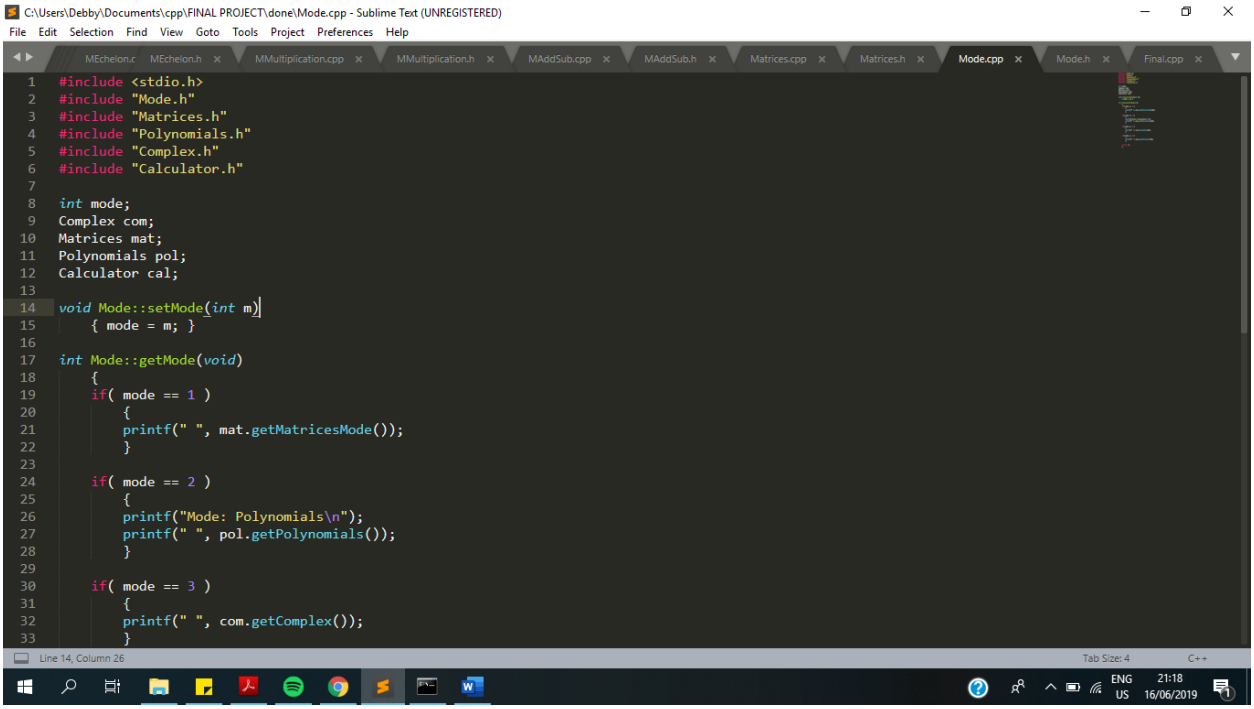
## Final.cpp



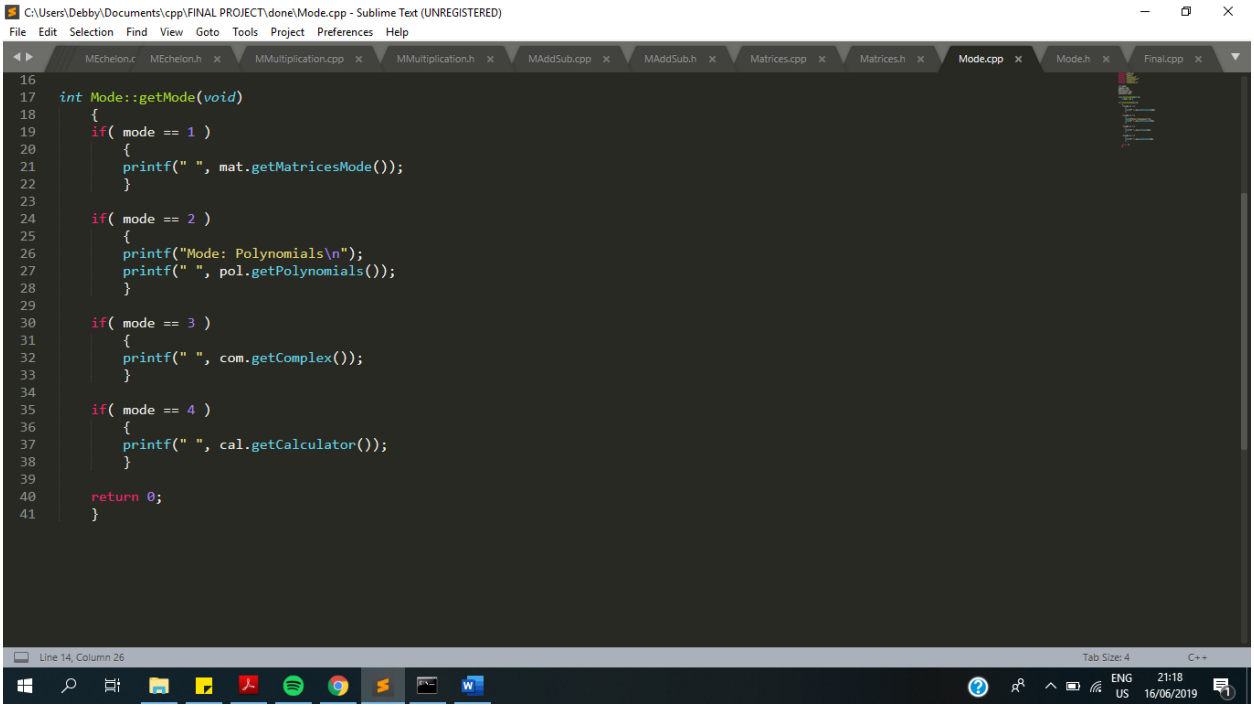

## Mode.h



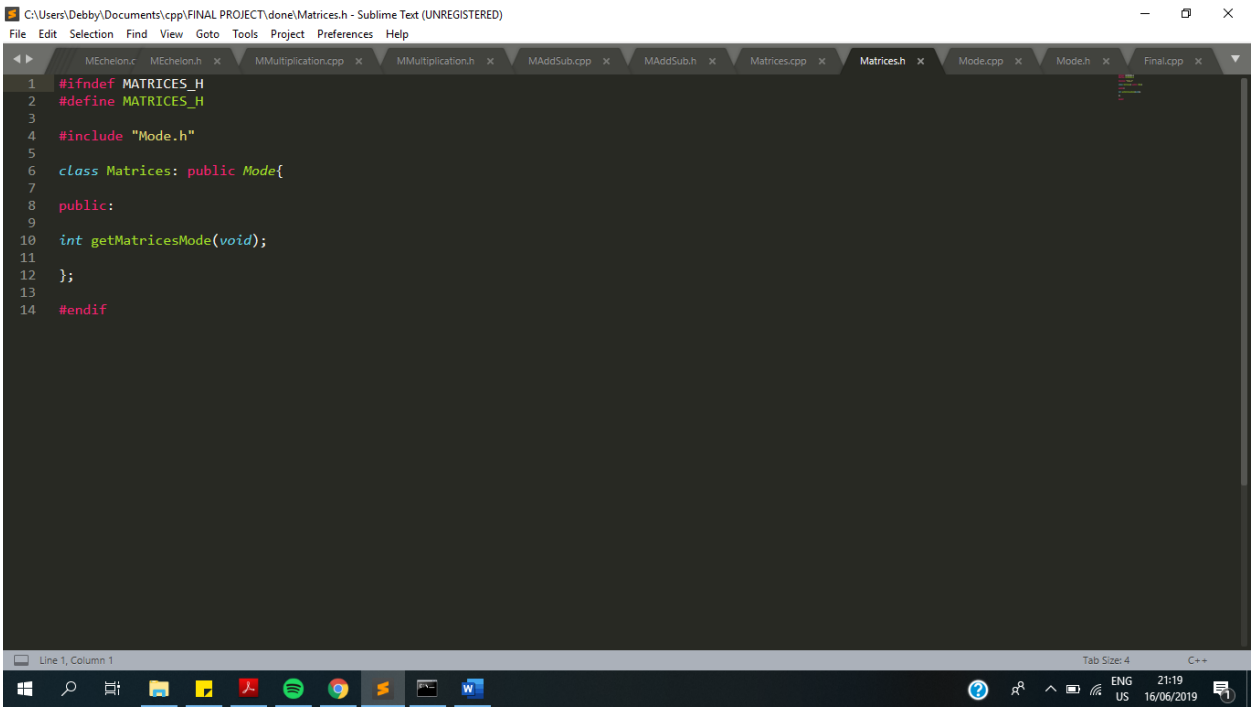
## Mode.cpp (I)



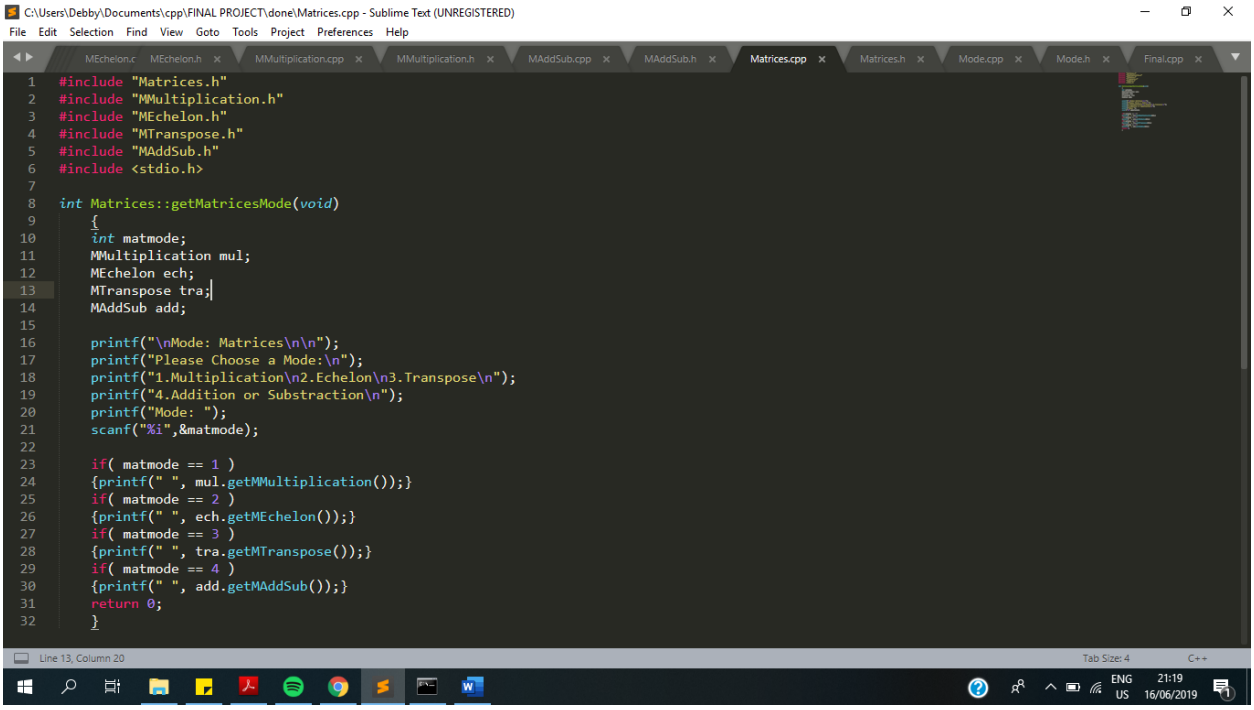

## Mode.cpp (II)



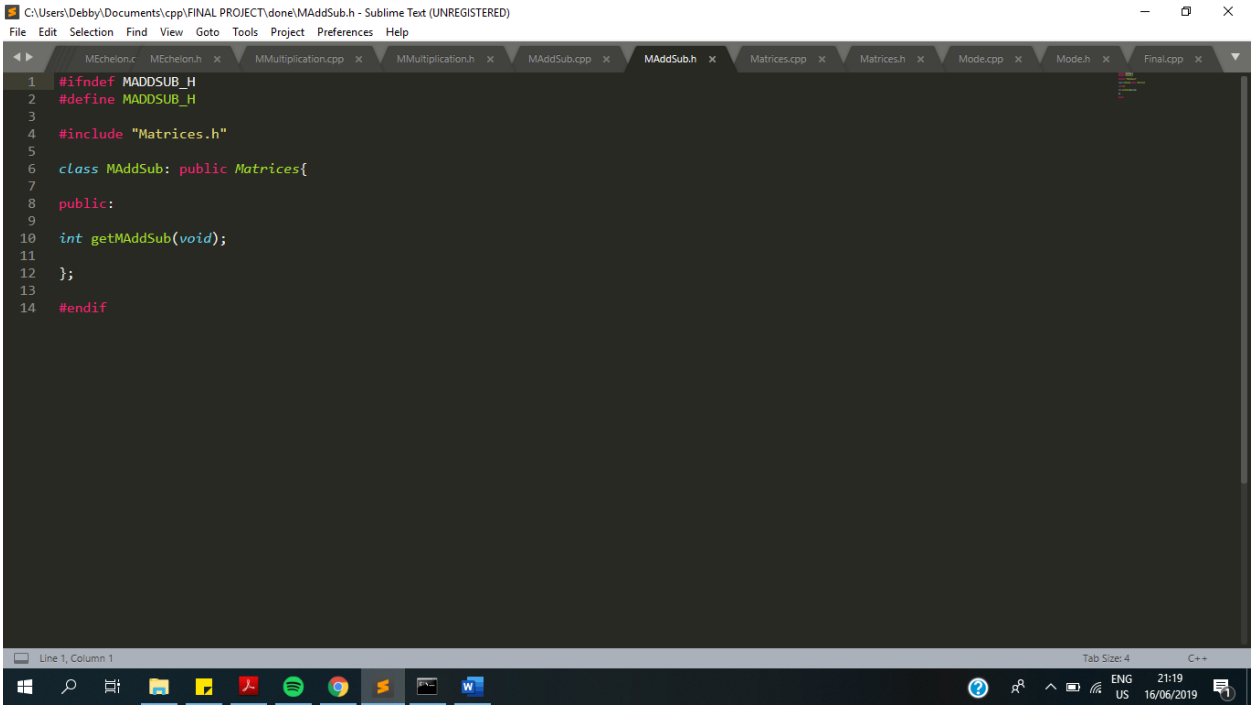
## Matrices.h



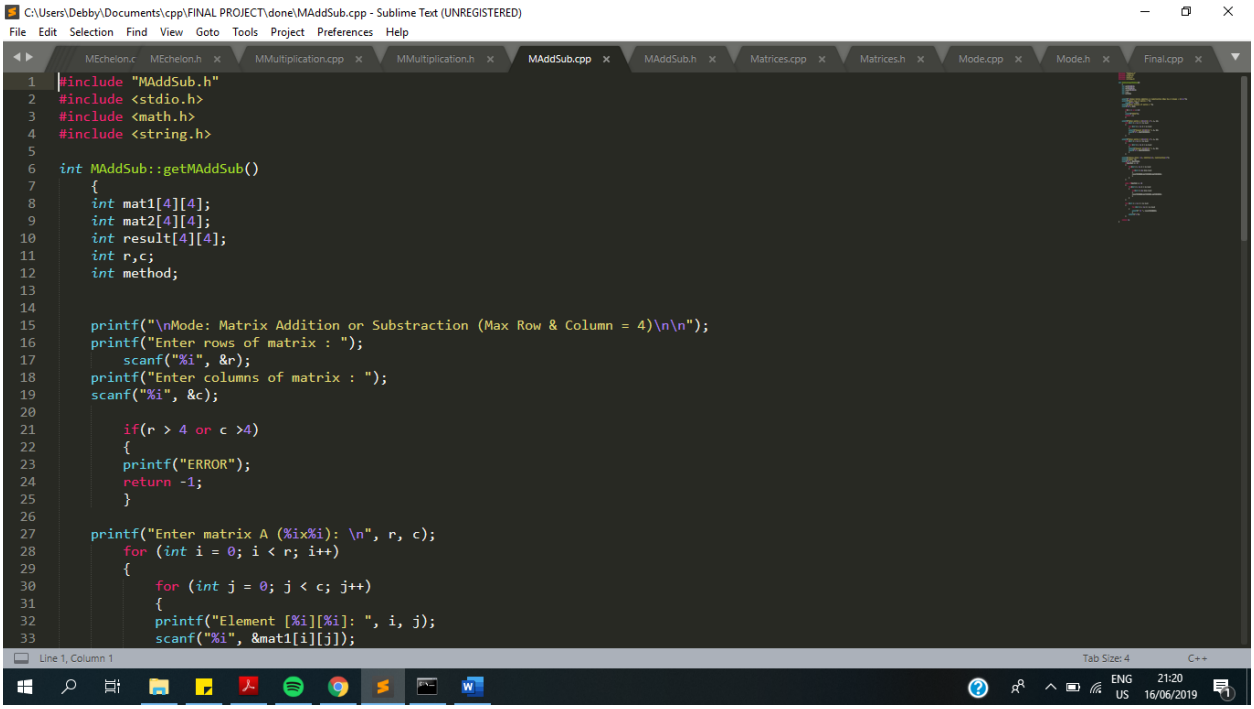

## Matrices.cpp



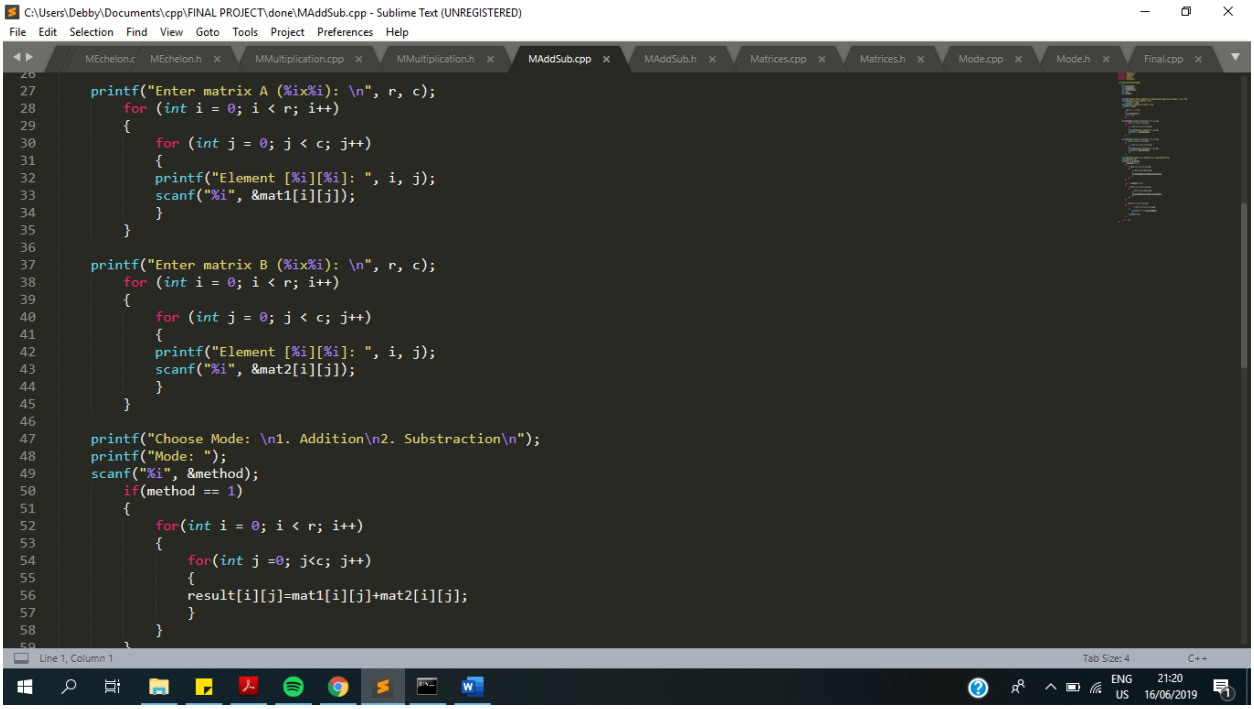
## MAddSub.h



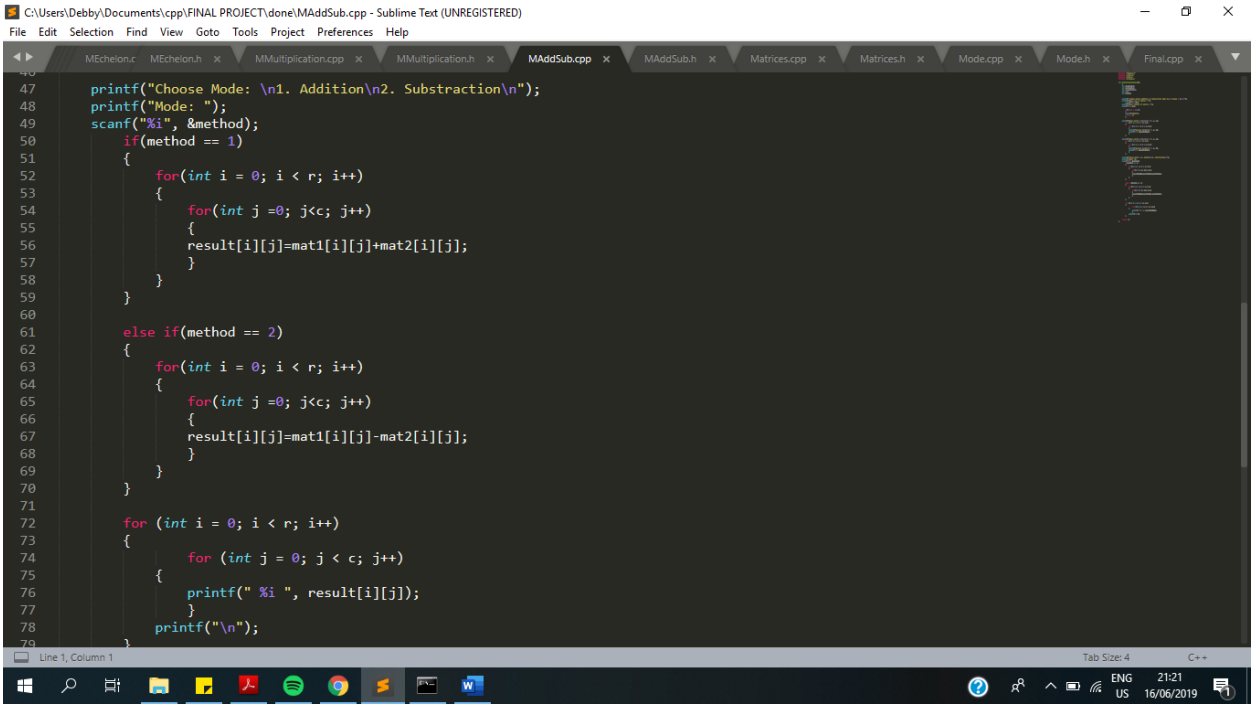

## MAddSub.cpp (I)



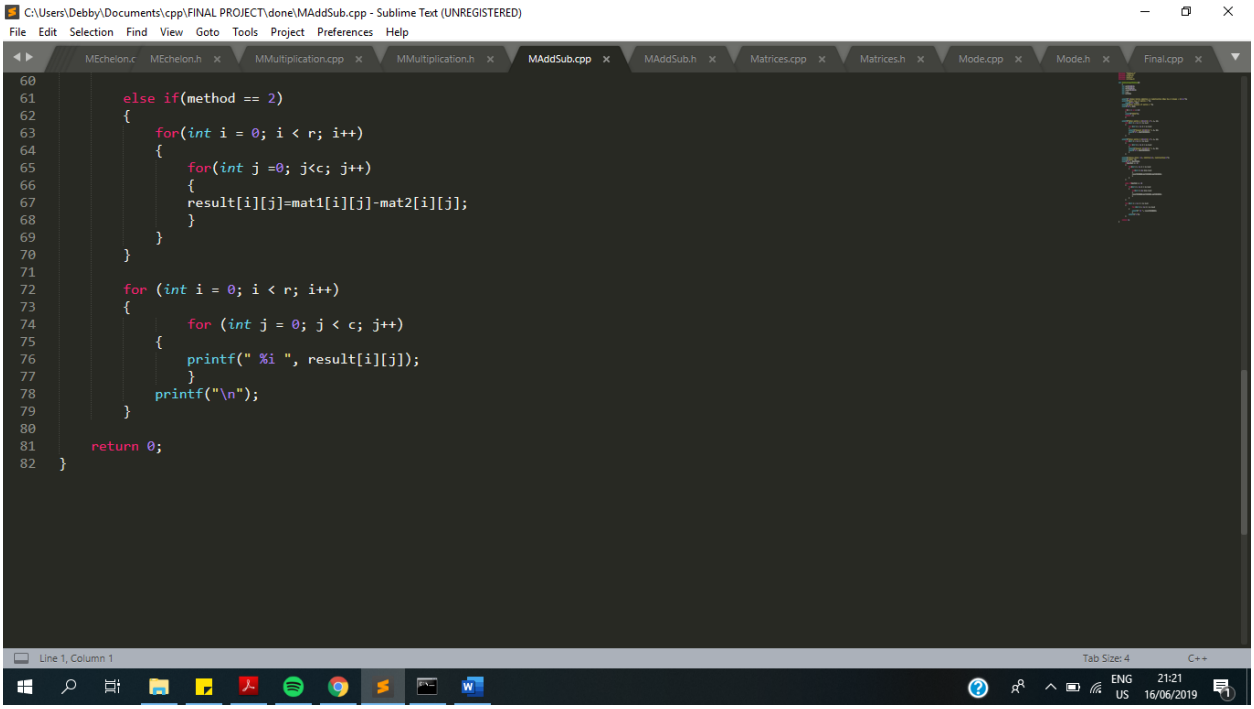
## MAddSub.cpp (II)



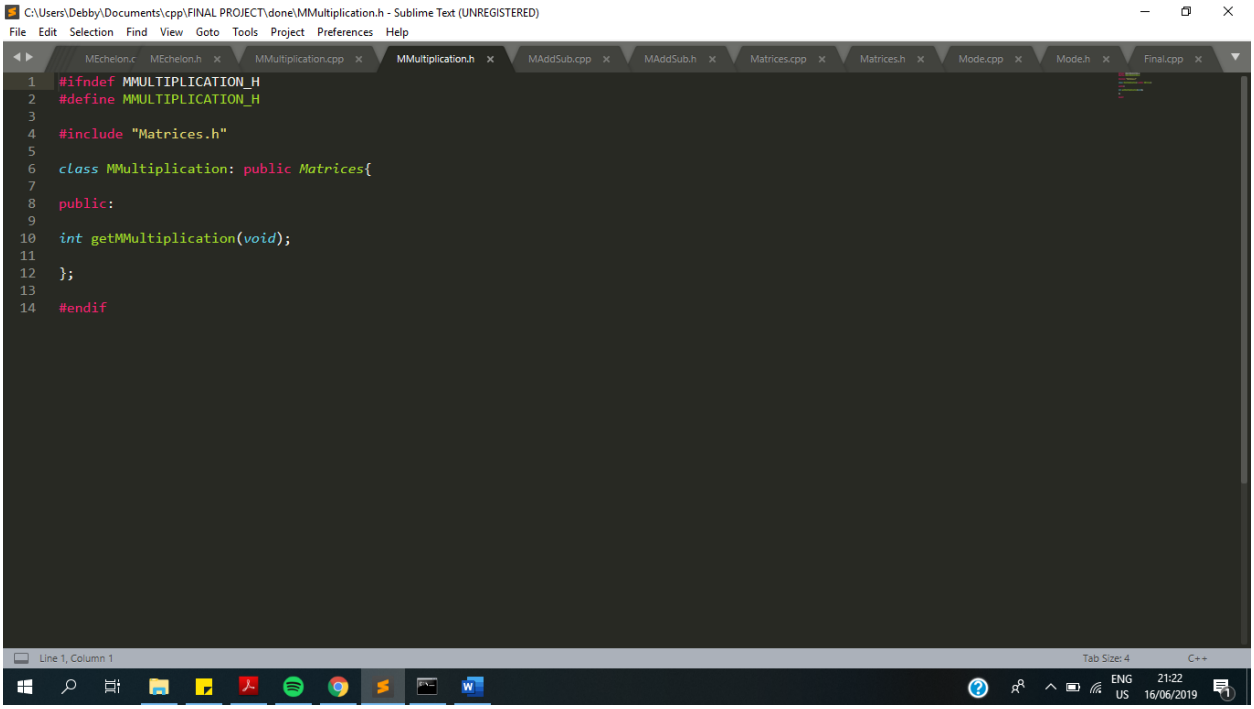

## MAddSub.cpp (III)



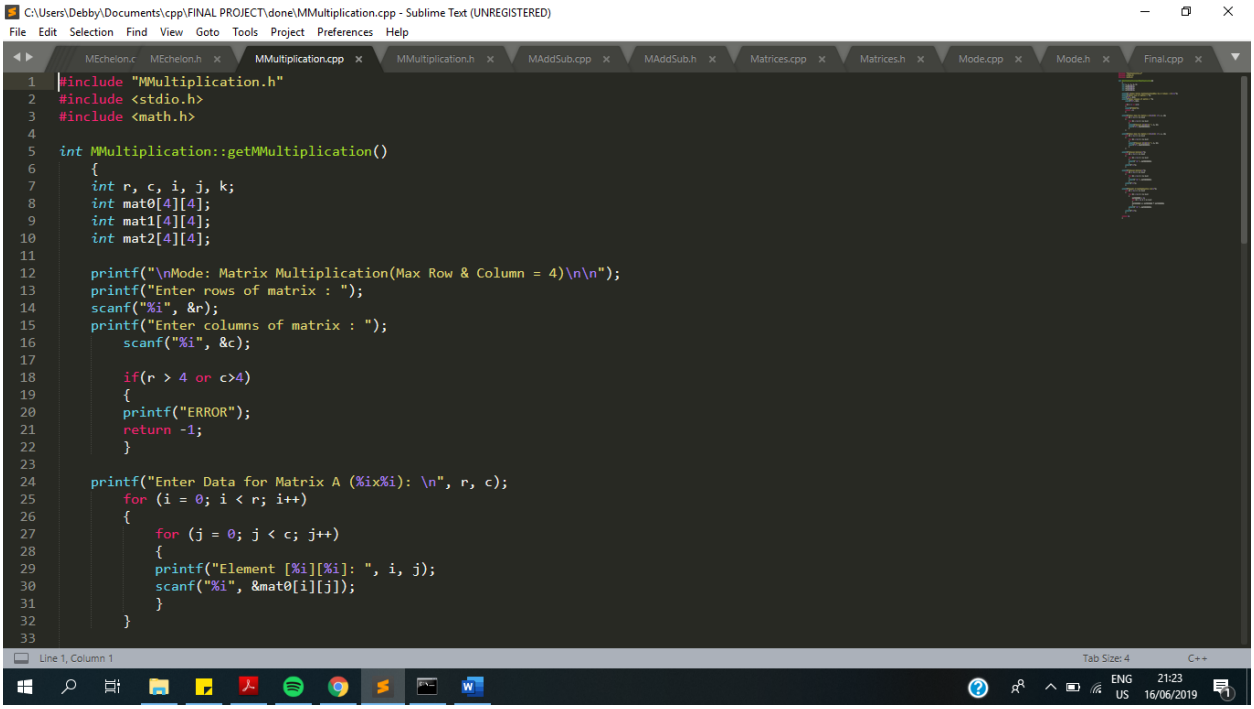
## MAddSub.cpp (IV)



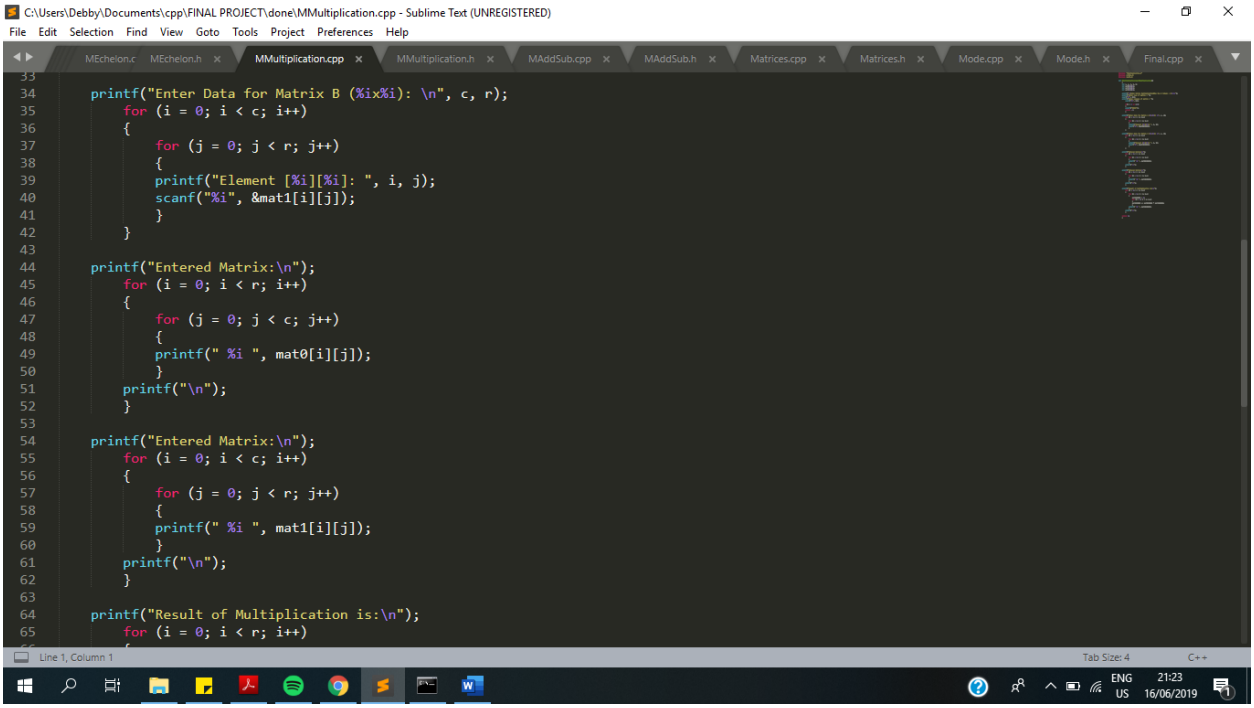

## MMultiplication.h



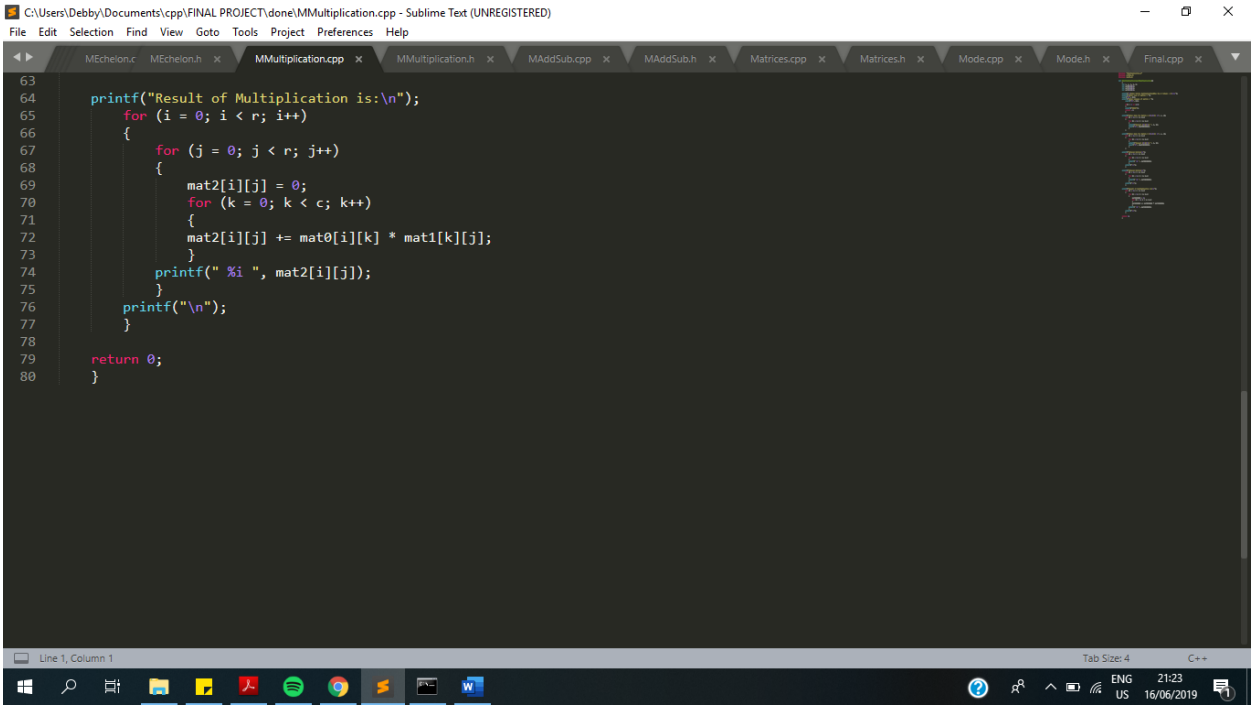
## MMultiplication.cpp (I)



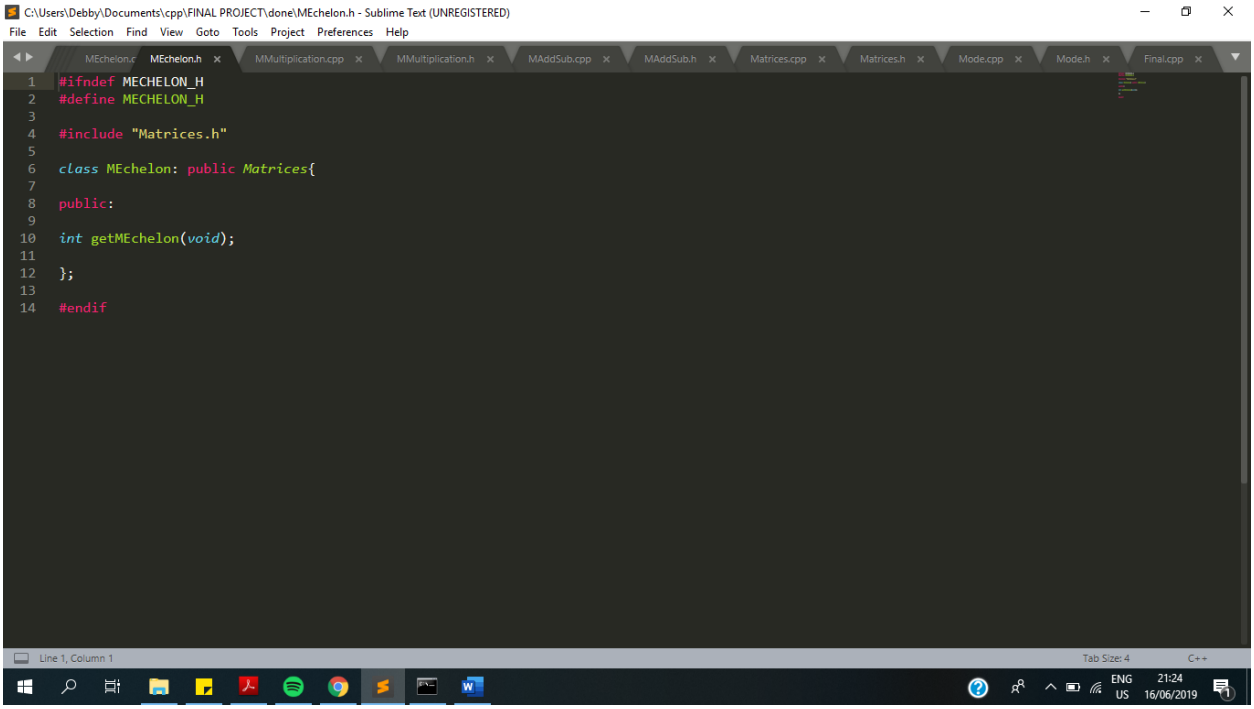

## MMultiplication.cpp (II)



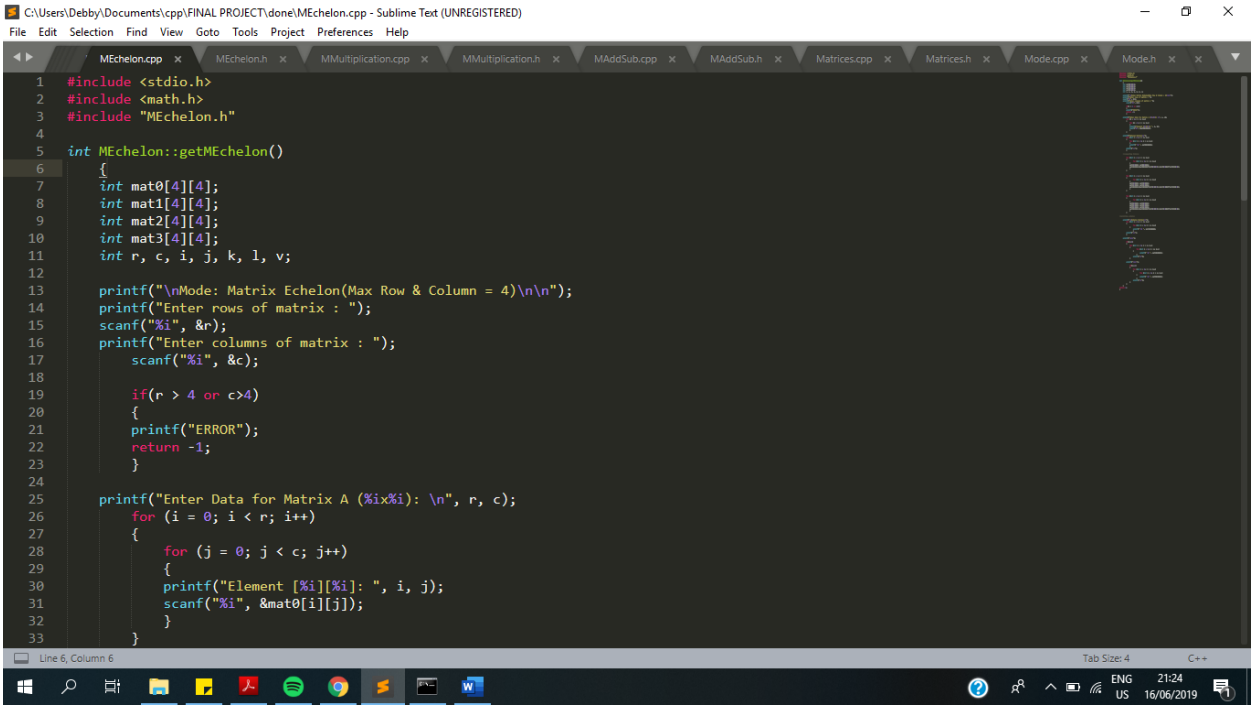
## MMultiplication.cpp (III)




## MEchelon.h

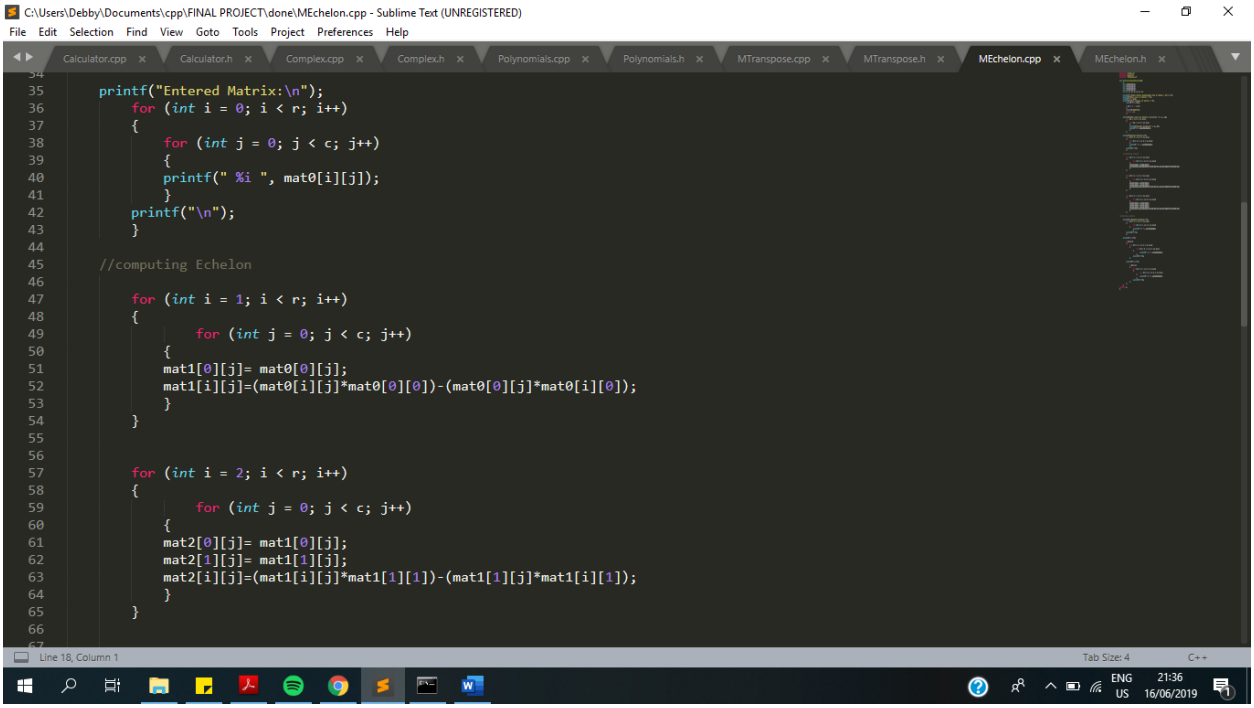


## MEchelon.cpp (I)



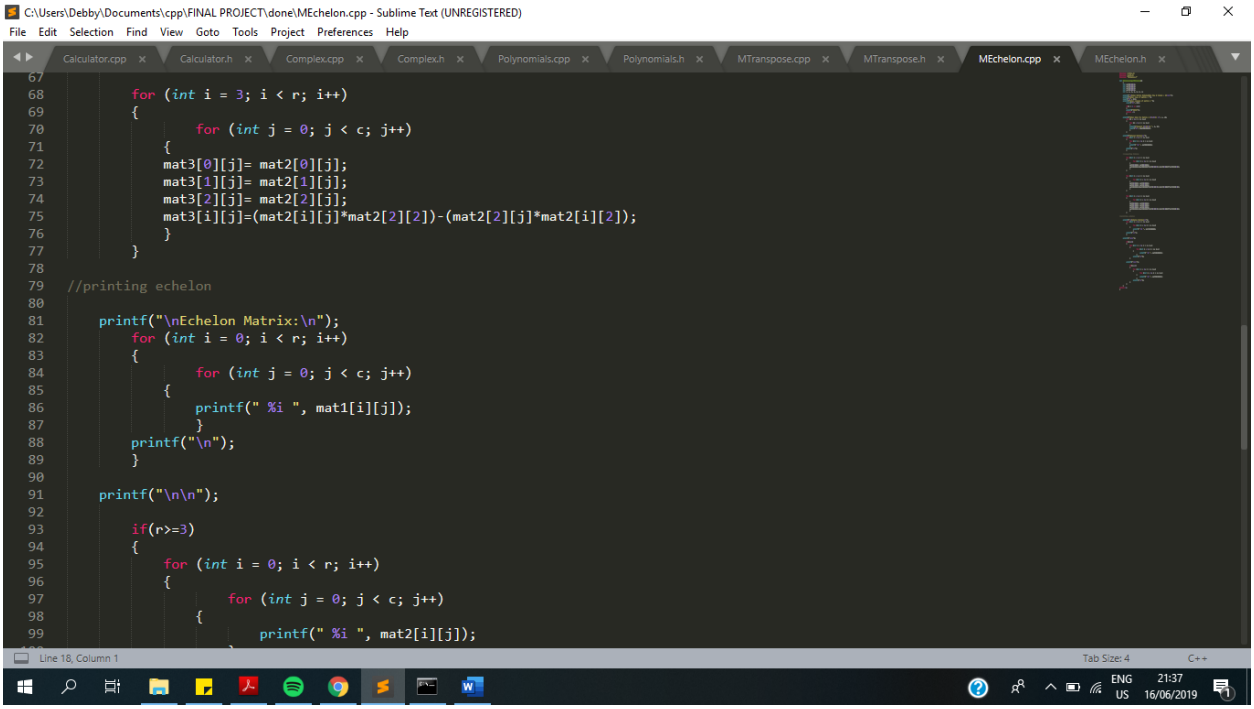
```
#include <stdio.h>
#include <math.h>
#include "MEchelon.h"

int MEchelon::getMEchelon()
    {
    int mat0[4][4];
    int mat1[4][4];
    int mat2[4][4];
    int mat3[4][4];
    int r, c, i, j, k, l, v;

    printf("\nMode: Matrix Echelon(Max Row & Column = 4)\n\n");
    printf("Enter rows of matrix : ");
    scanf("%i", &r);
    printf("Enter columns of matrix : ");
        scanf("%i", &c);

        if(r > 4 or c>4)
        {
        printf("ERROR");
        return -1;
        }

    printf("Enter Data for Matrix A (%ix%i): \n", r, c);
        for (i = 0; i < r; i++)
        {
            for (j = 0; j < c; j++)
            {
            printf("Element [%i][%i]: ", i, j);
            scanf("%i", &mat0[i][j]);
            }
        }
```

## MEchelon.cpp (II)



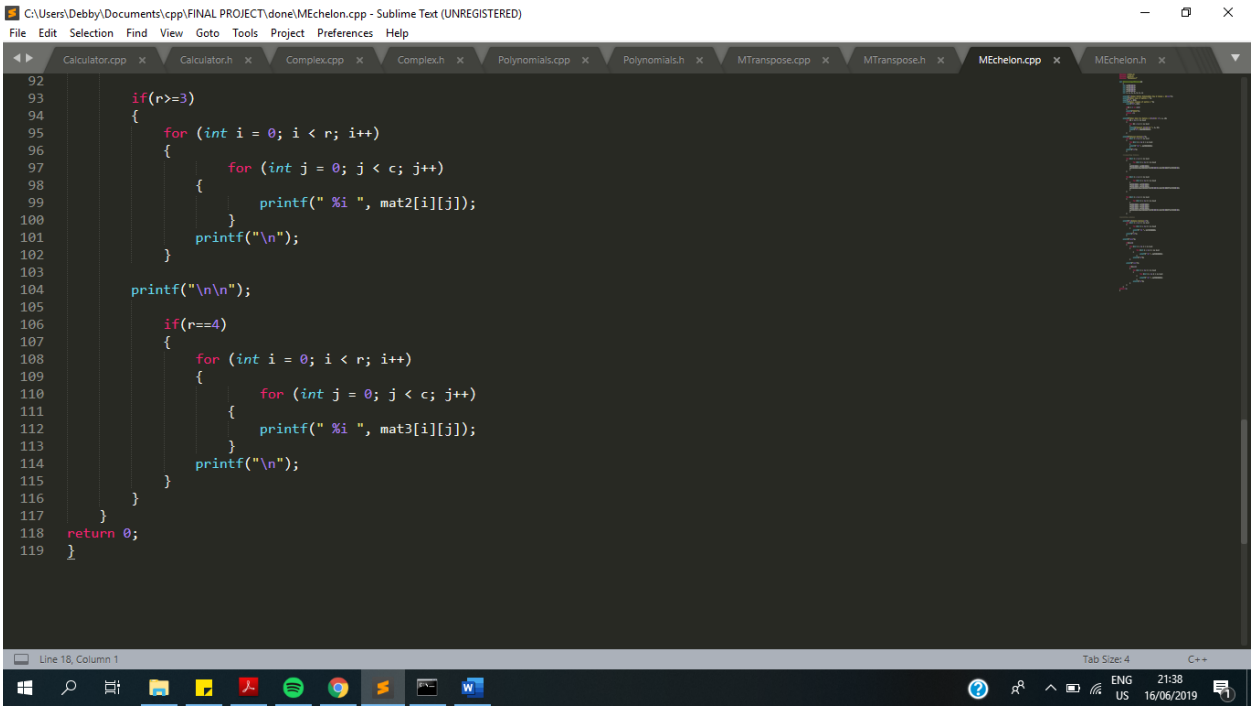
```
    printf("Entered Matrix:\n");
        for (int i = 0; i < r; i++)
        {
            for (int j = 0; j < c; j++)
            {
            printf(" %i ", mat0[i][j]);
            }
        printf("\n");
        }

    //computing Echelon

        for (int i = 1; i < r; i++)
        {
                for (int j = 0; j < c; j++)
            {
            mat1[0][j]= mat0[0][j];
            mat1[i][j]=(mat0[i][j]*mat0[0][0])-(mat0[0][j]*mat0[i][0]);
            }
        }


        for (int i = 2; i < r; i++)
        {
                for (int j = 0; j < c; j++)
            {
            mat2[0][j]= mat1[0][j];
            mat2[1][j]= mat1[1][j];
            mat2[i][j]=(mat1[i][j]*mat1[1][1])-(mat1[1][j]*mat1[i][1]);
            }
        }
```

## MEchelon.cpp (III)



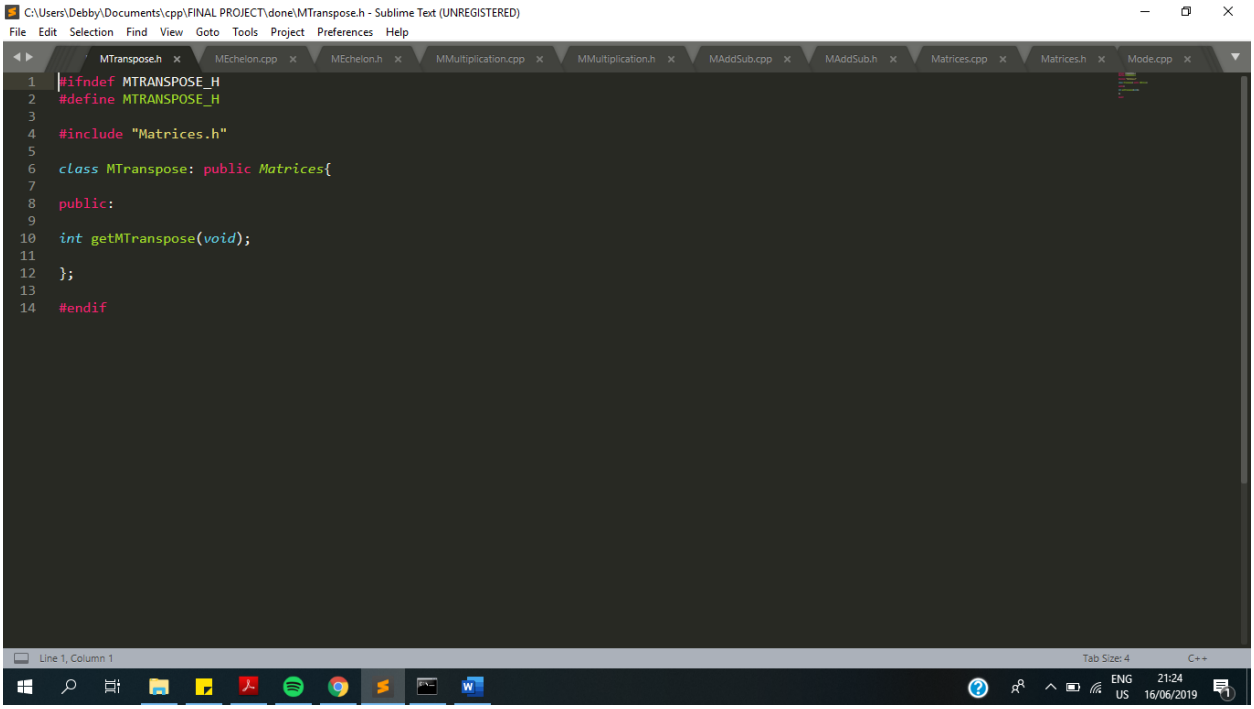
```
        for (int i = 3; i < r; i++)
        {
                for (int j = 0; j < c; j++)
            {
            mat3[0][j]= mat2[0][j];
            mat3[1][j]= mat2[1][j];
            mat3[2][j]= mat2[2][j];
            mat3[i][j]=(mat2[i][j]*mat2[2][2])-(mat2[2][j]*mat2[i][2]);
            }
        }

//printing echelon

    printf("\nEchelon Matrix:\n");
        for (int i = 0; i < r; i++)
        {
                for (int j = 0; j < c; j++)
            {
                printf(" %i ", mat1[i][j]);
                }
        printf("\n");
        }

    printf("\n\n");

        if(r>=3)
        {
            for (int i = 0; i < r; i++)
            {
                    for (int j = 0; j < c; j++)
                {
                    printf(" %i ", mat2[i][j]);
```

## MEchelon.cpp (IV)

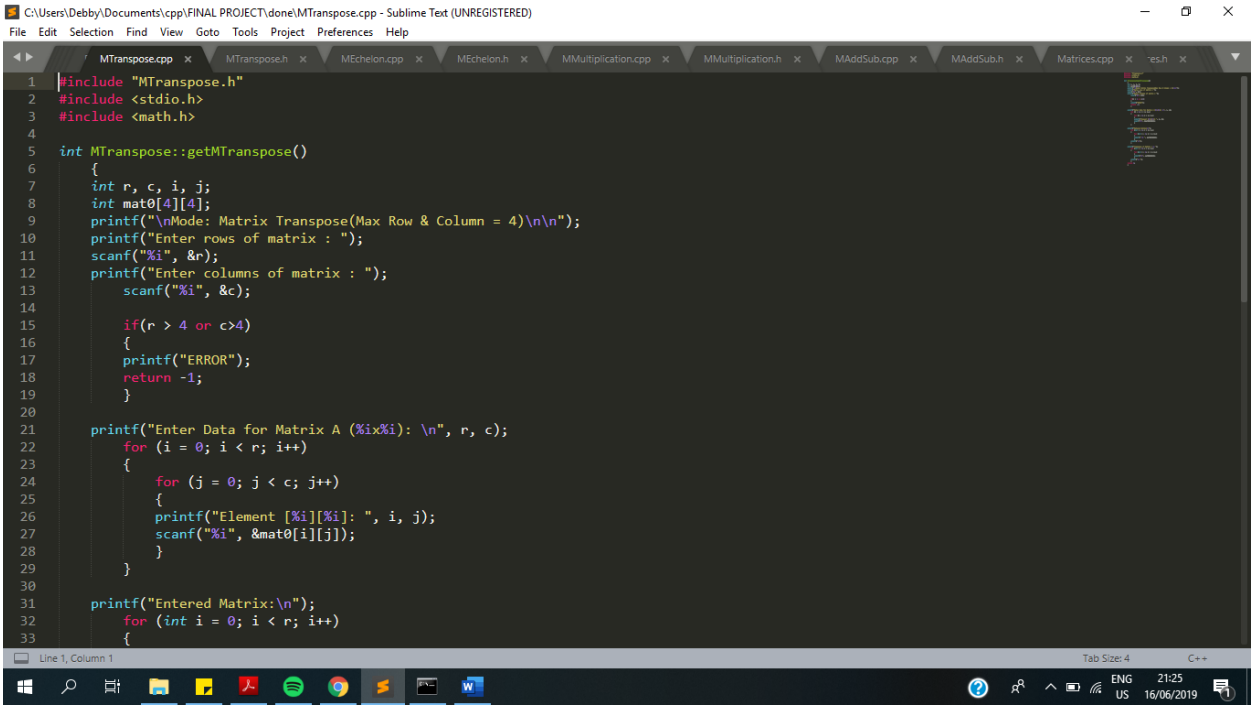


```
        if(r>=3)
        {
            for (int i = 0; i < r; i++)
            {
                    for (int j = 0; j < c; j++)
                {
                    printf(" %i ", mat2[i][j]);
                }
            printf("\n");
            }

        printf("\n\n");

            if(r==4)
            {
                for (int i = 0; i < r; i++)
                {
                        for (int j = 0; j < c; j++)
                    {
                        printf(" %i ", mat3[i][j]);
                    }
                printf("\n");
                }
            }
    }
return 0;
}
```

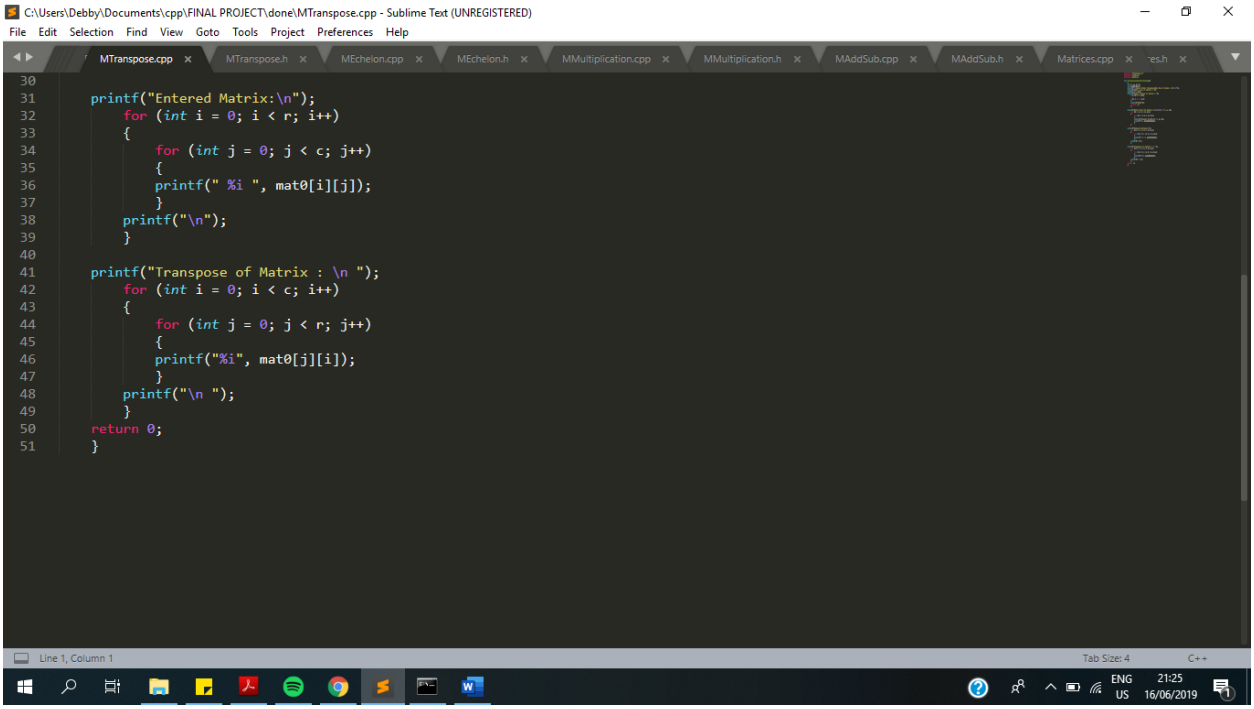
## MTranspose.h



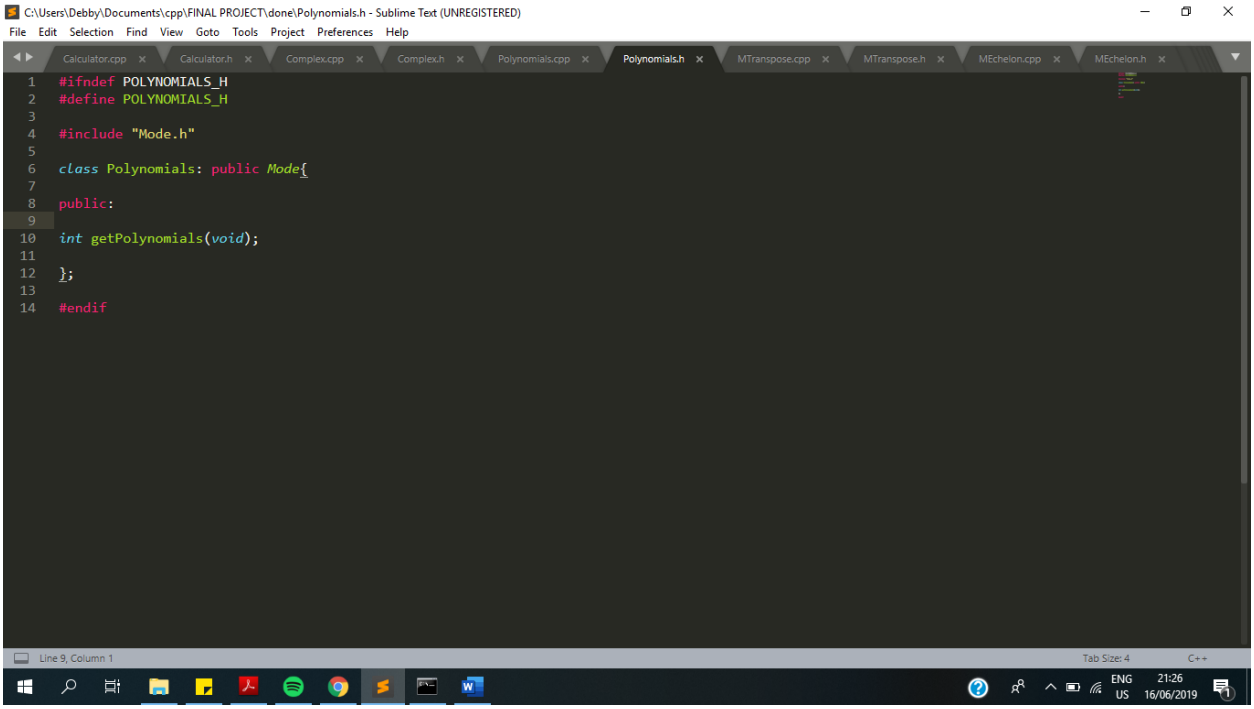

## MTranspose.cpp (I)



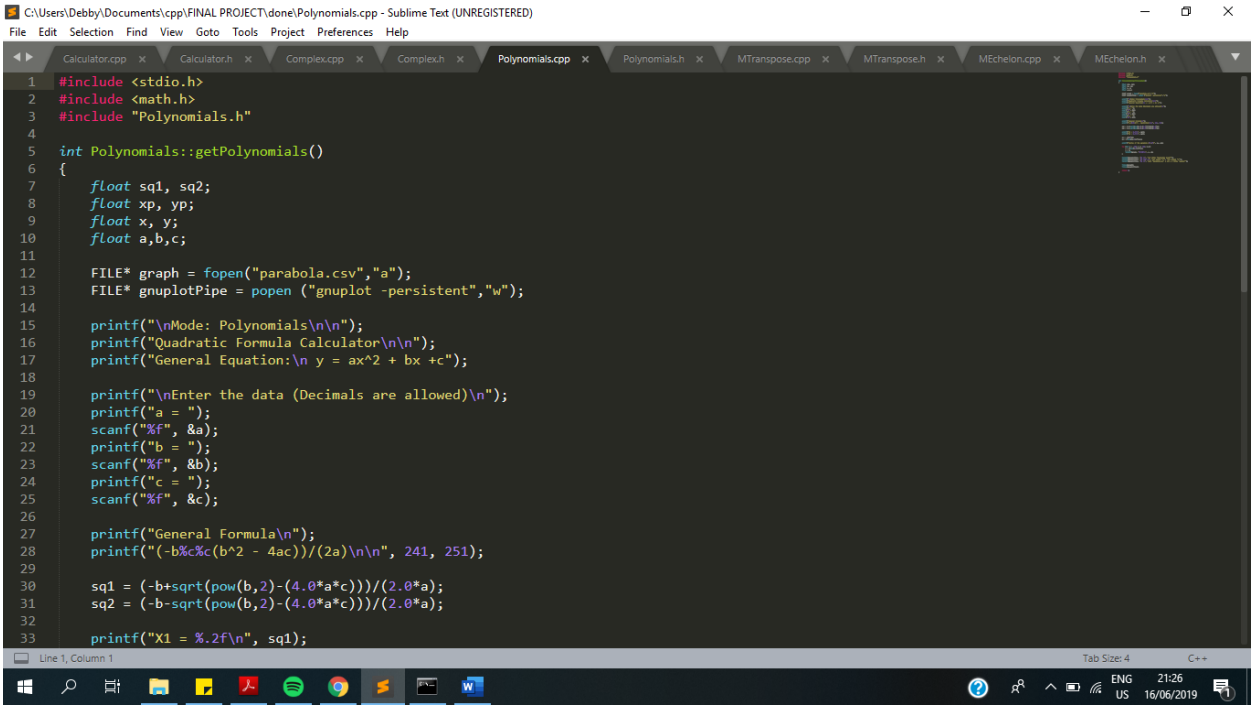
## MTranspose.cpp (II)




## Polynomials.h

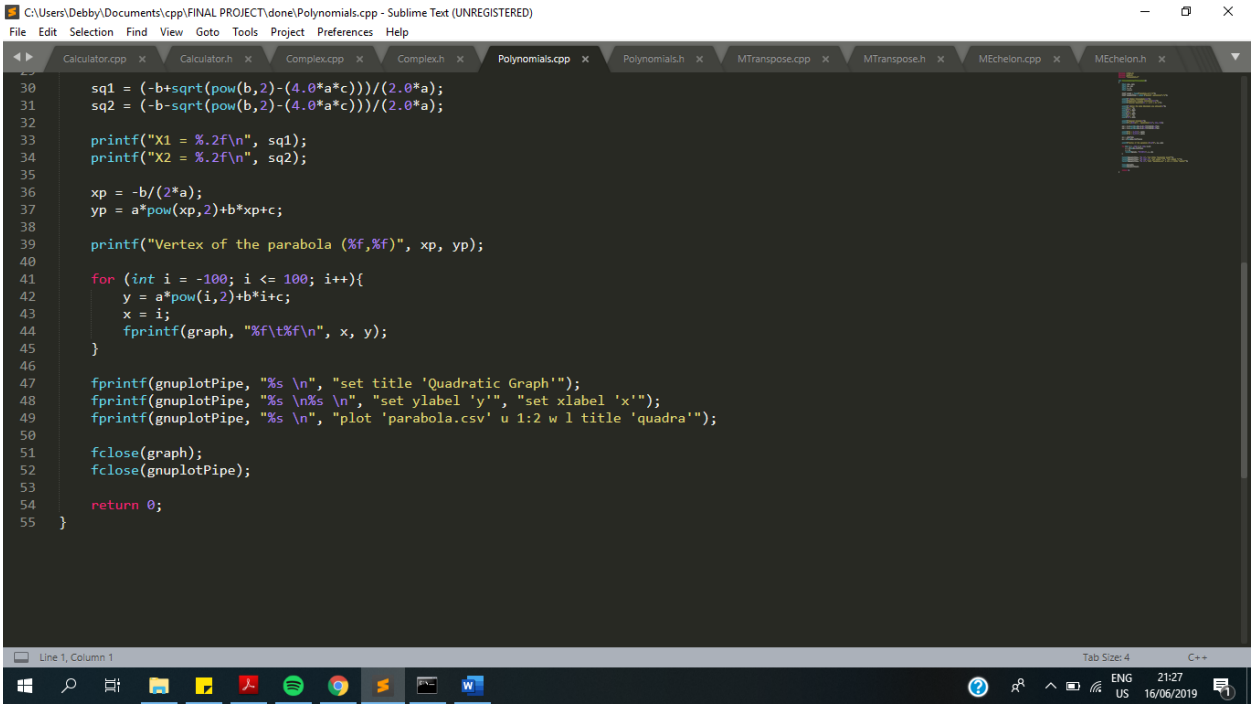


## Polynomials.cpp (I)



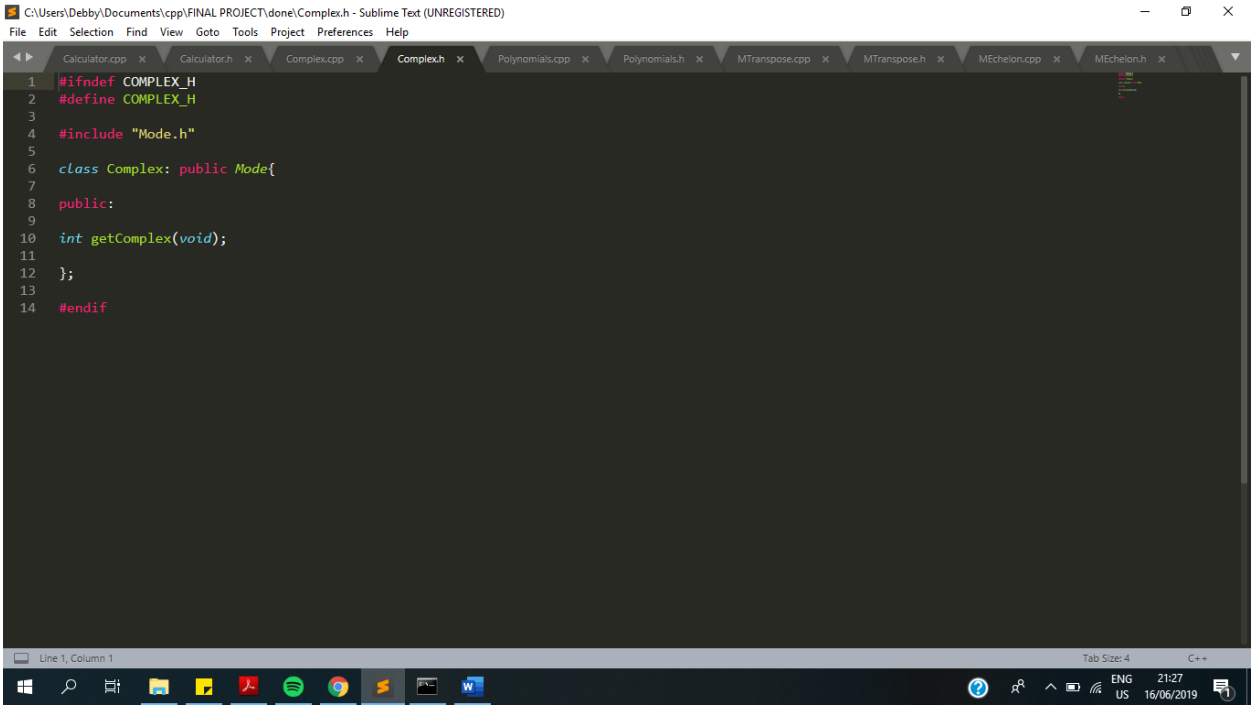

```cpp
#include <stdio.h>
#include <math.h>
#include "Polynomials.h"

int Polynomials::getPolynomials()
{
    float sq1, sq2;
    float xp, yp;
    float x, y;
    float a,b,c;

    FILE* graph = fopen("parabola.csv","a");
    FILE* gnuplotPipe = popen ("gnuplot -persistent","w");

    printf("\nMode: Polynomials\n\n");
    printf("Quadratic Formula Calculator\n\n");
    printf("General Equation:\n y = ax^2 + bx +c");

    printf("\nEnter the data (Decimals are allowed)\n");
    printf("a = ");
    scanf("%f", &a);
    printf("b = ");
    scanf("%f", &b);
    printf("c = ");
    scanf("%f", &c);

    printf("General Formula\n");
    printf("(-b%c%c(b^2 - 4ac))/(2a)\n\n", 241, 251);

    sq1 = (-b+sqrt(pow(b,2)-(4.0*a*c)))/(2.0*a);
    sq2 = (-b-sqrt(pow(b,2)-(4.0*a*c)))/(2.0*a);

    printf("X1 = %.2f\n", sq1);
```

## Polynomials.cpp (II)

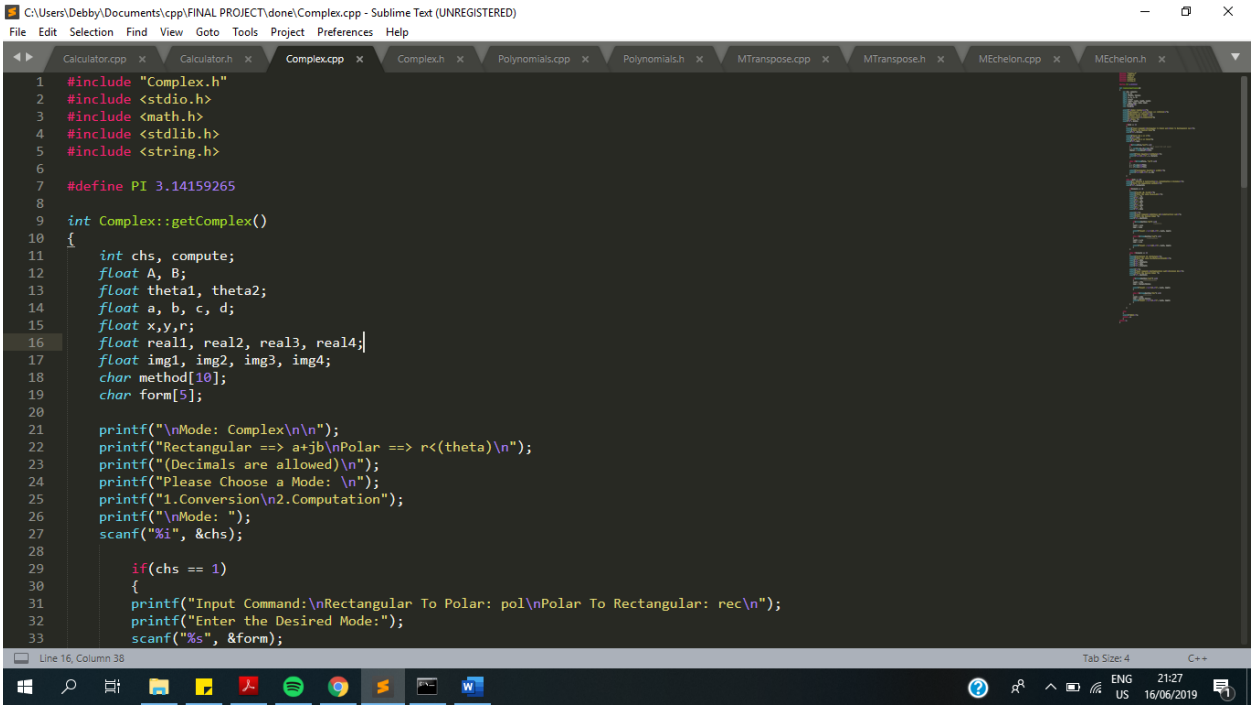



```cpp
    sq1 = (-b+sqrt(pow(b,2)-(4.0*a*c)))/(2.0*a);
    sq2 = (-b-sqrt(pow(b,2)-(4.0*a*c)))/(2.0*a);

    printf("X1 = %.2f\n", sq1);
    printf("X2 = %.2f\n", sq2);

    xp = -b/(2*a);
    yp = a*pow(xp,2)+b*xp+c;

    printf("Vertex of the parabola (%f,%f)", xp, yp);

    for (int i = -100; i <= 100; i++){
        y = a*pow(i,2)+b*i+c;
        x = i;
        fprintf(graph, "%f\t%f\n", x, y);
    }

    fprintf(gnuplotPipe, "%s \n", "set title 'Quadratic Graph'");
    fprintf(gnuplotPipe, "%s \n%s \n", "set ylabel 'y'", "set xlabel 'x'");
    fprintf(gnuplotPipe, "%s \n", "plot 'parabola.csv' u 1:2 w l title 'quadra'");

    fclose(graph);
    fclose(gnuplotPipe);

    return 0;
}
```

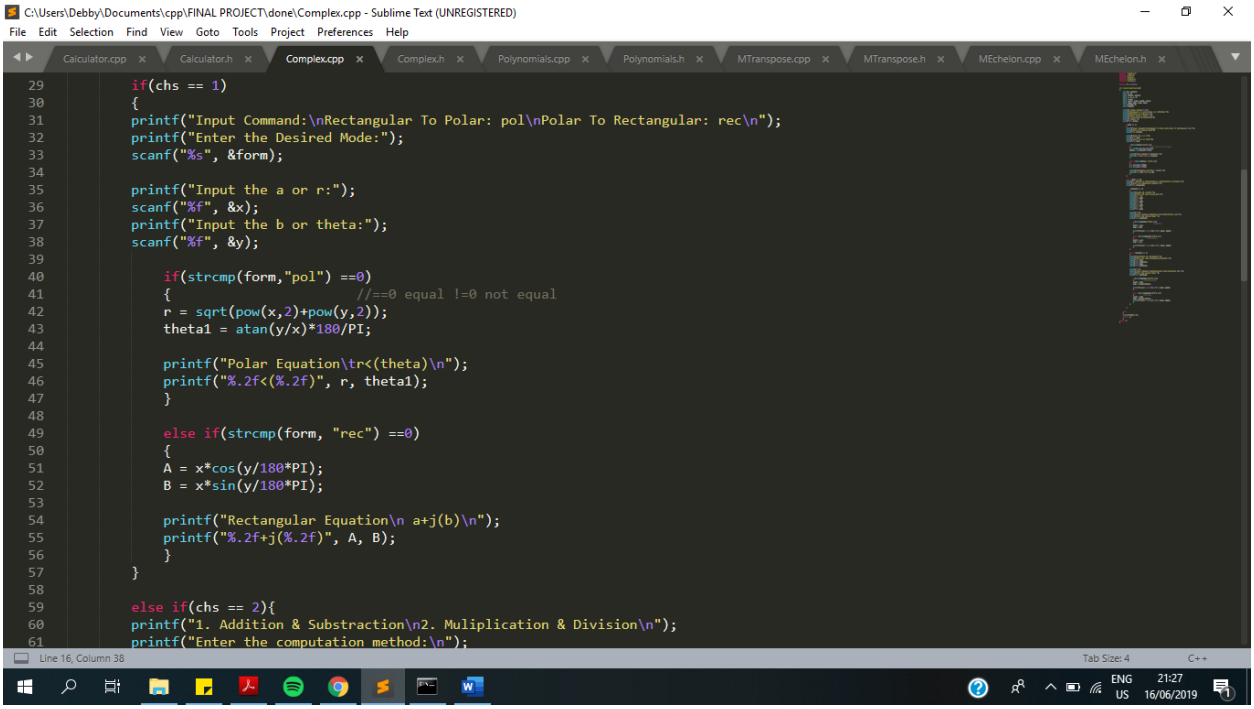
## Complex.h



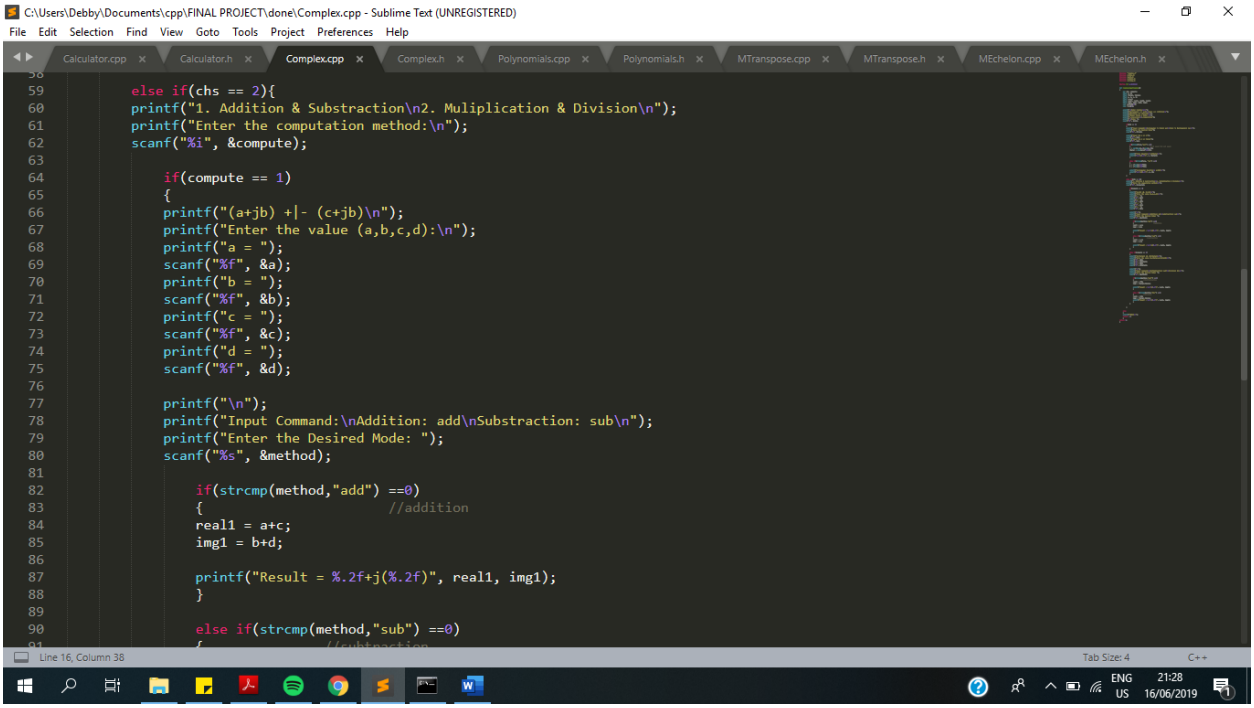

## Complex.cpp (I)



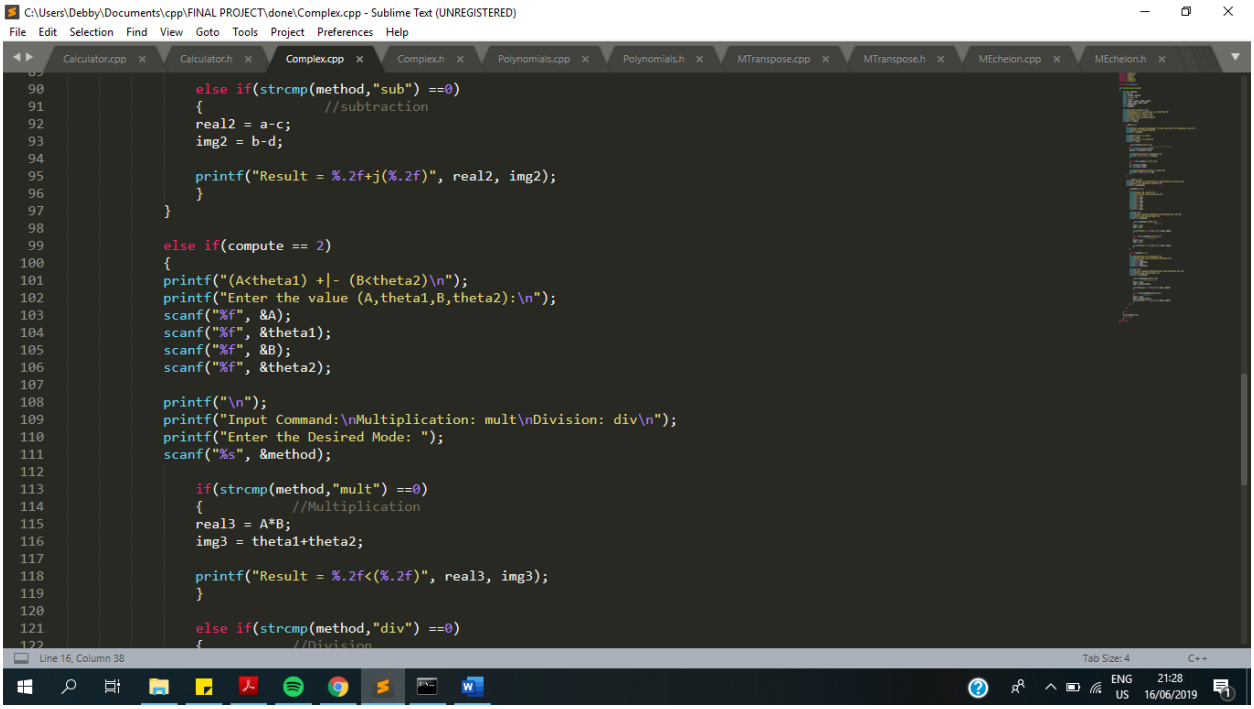
## Complex.cpp (II)



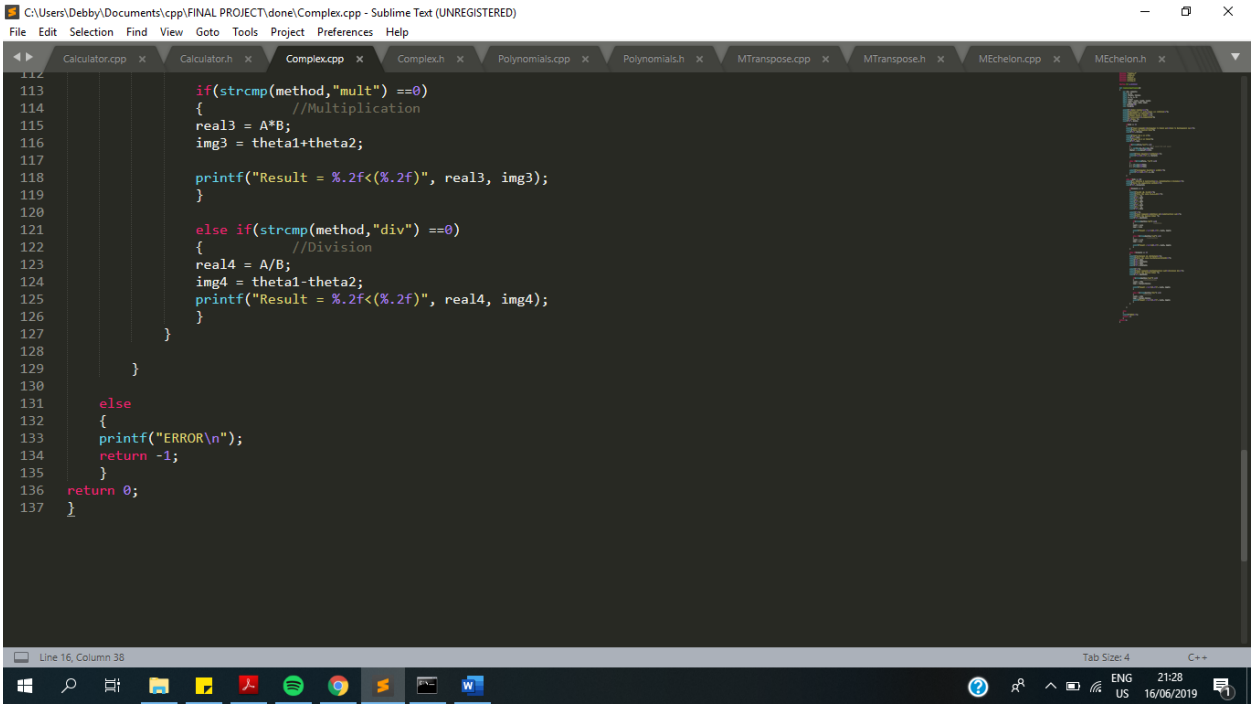

## Complex.cpp (III)



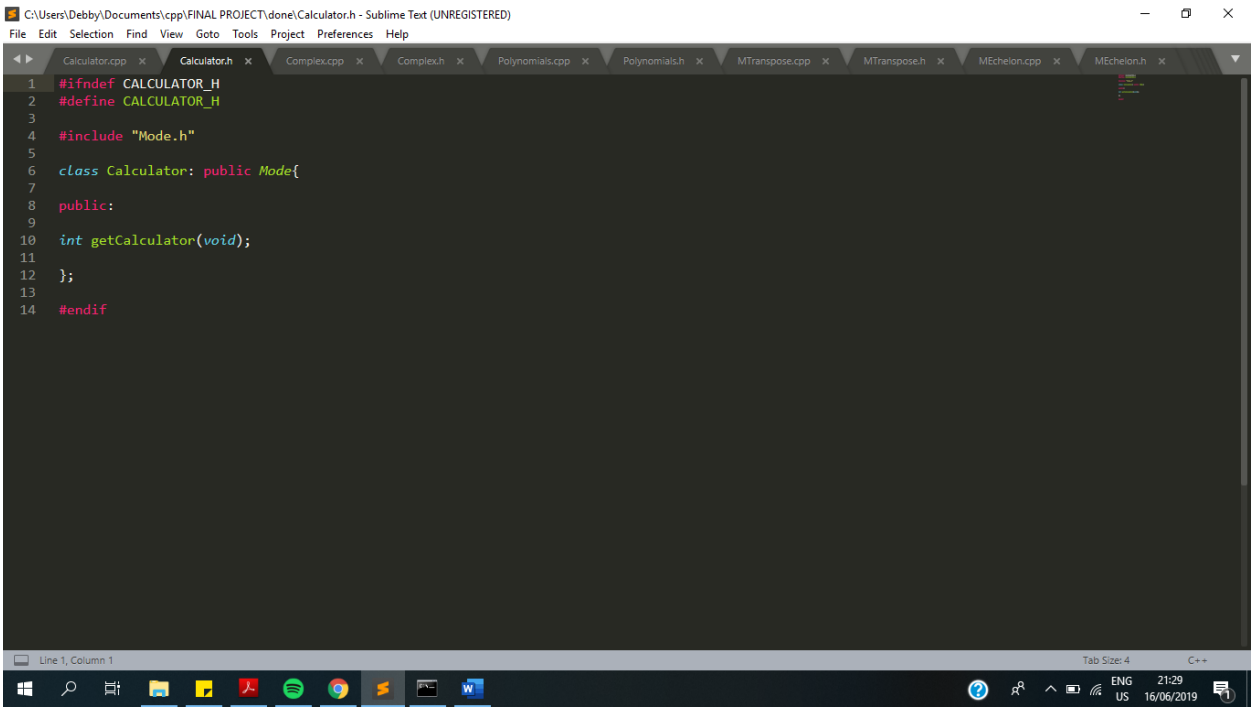
## Complex.cpp (IV)



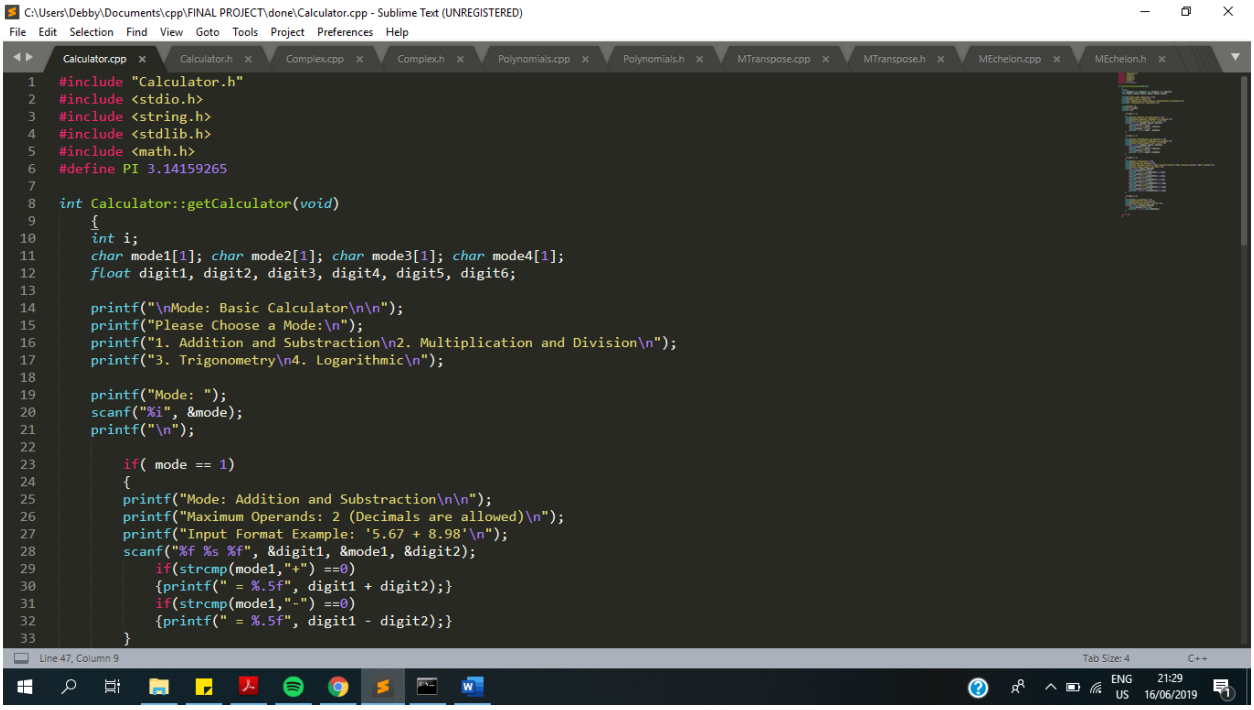

## Complex.cpp (V)



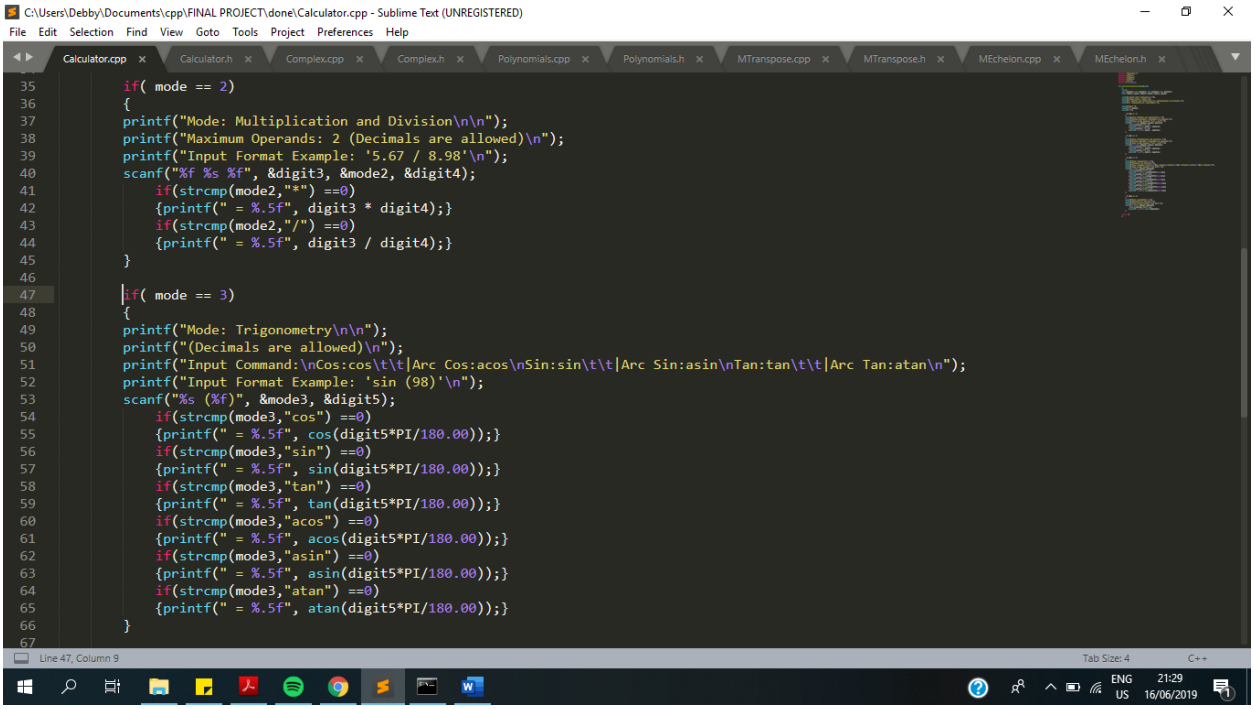
## Calculator.h




## Calculator.cpp (I)



## Calculator.cpp (II)




## Calculator.cpp (III)