JAMIA MILLIA ISLAMIA, NEW DELHI

MATHEMATICS-IV (NACP) LAB

NAME: FAIZAN CHOUDHARY

ROLL NO: 20BCS021

SUBJECT CODE: CEN 401

SEMESTER: 4th

COURSE: B.TECH.(COMPUTER ENGG.)

DEPT: DEPT OF COMPUTER ENGG.

PROGRAM 1

Newton's Divided Difference Interpolation

CODE:

```cpp
#include <iostream>
using namespace std;
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int n,i, j=1;
    double f, f1=1, f2=0, k;
    cout<<"\nEnter the number of observations: ";
    cin>>n;
    int *x = new int[n+1];
    int *y = new int[n+1];
    int *p = new int[n+1];
    cout<<"\nEnter the values of x: ";
    for(i=1;i<=n;i++)
        cin>>x[i];
    cout<<"\nEnter corresponding values of y: ";
    for(i=1;i<=n;i++)
        cin>>y[i];
    f = y[1];
    cout<<"\nEnter the value of k for which f(k) is to be calculated: ";
    cin>>k;
    do {
        for (i=1; i<=n-1; i++) {
            p[i] = ((y[i+1]-y[i])/(x[i+j]-x[i]));
            y[i] = p[i];
        }
        f1 = 1;
        for (i=1; i<=j; i++)
            f1 *= (k - x[i]);
        f2 += (y[1] * f1);
        j++;
        n--;
    } while (n != 1);
    f += f2;
    cout<<"\nThe value of f("<<k<<") is: "<<f<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of observations: 5

Enter the values of x: 5 7 11 13 17

Enter corresponding values of y: 150 392 1452 2366 5202

Enter the value of k for which f(k) is to be calculated: 7.5

The value of f(7.5) is: 478.125
```

PROGRAM 2

Lagrange's Interpolation

CODE:

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

int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int i, j, n;
    double nr, dr, x, y=0.0, ax[15], ay[15];

    cout<<"\nEnter the number of points: ";
    cin>>n;

    cout<<"\nEnter the values of x: ";
    for(i=0;i<n;i++)
        cin>>ax[i];

    cout<<"\nEnter corresponding values of y: ";
    for(i=0;i<n;i++)
        cin>>ay[i];

    cout<<"\nEnter the value of x: ";
    cin>>x;

    for(i=0; i<n ;i++) {
        nr = dr = 1;
        for(j=0; j<n ;j++) {
            if(j != i) {
                nr *= (x - ax[j]);
                dr *= (ax[i] - ax[j]);
            }
        }
        y += (nr/dr)*ay[i];
    }
    cout<<"\nThe value of y("<<x<<") is: "<<y<<endl;

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of points: 5

Enter the values of x: 5 7 11 13 17

Enter corresponding values of y: 150 392 1452 2366 5202

Enter the value of x: 9

The value of y(9) is: 810
```

PROGRAM 3

Boole's Rule

CODE:

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
double f (double a) {
    // integral of (2-2x+sin(x-1)+x^2)/(1+(x-1)^2)
    double ans;
    ans = (2 - 2*a + sin(a-1) + a*a) / (1 + (a-1)*(a-1));
    return ans;
}
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int i, n;
    double x[15], y[15], k1, k2, k3, h, boole;
    x[0] = 0.5;
    n=12;
    x[n] = 1.7;
    k1 = k2 = k3 = 0.0;
    h = (x[n] - x[0]) / n;
    for (i=1; i<n; i++)
        x[i] = x[0] + i*h;
    for (i=0; i<=n; i++)
        y[i] = f(x[i]);
    for (i=1; i<n; i++) {
        if (i%4 == 0)
            k1 += 14 * y[i];
        else {
            if (i%2 == 0)
                k2 += 12 * y[i];
            else
                k3 += 32 * y[i];
        }
    }
    cout<<"\nThe values of x are: ";
    for (i=0; i<=n; i++)
        cout<<setprecision(4)<<fixed<<x[i]<<" ";
    cout<<"\nThe values of y are: ";
    for (i=0; i<=n; i++)
        cout<<setprecision(4)<<fixed<<y[i]<<" ";
    cout<<endl;
    boole = ((2*h) / 45) * (7*y[0] + 7*y[n] + k1 + k2 + k3);
    cout<<"\nThe value of the integral using Boole's rule:
"<<setprecision(4)<<fixed<<boole<<endl;
    return 0;
}
```

OUTPUT:

For the integral $\int_{0.5}^{1.7} \frac{2-2x+\sin(x-1)+x^2}{1+(x-1)^2}\,dx$ n=12

```
FAIZAN CHOUDHARY
20BCS021

The values of x are: 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000 1.1000 1.2000 1.3000 1.4000 1.5000 1.6000 1.7000

The values of y are: 0.6165 0.6643 0.7289 0.8090 0.9012 1.0000 1.0988 1.1910 1.2711 1.3357 1.3835 1.4152 1.4324

The value of the integral using Boole's rule: 1.2826
```

PROGRAM 4

Simpson's Rule

CODE:

```
//Simpson 1-3 Method for the evaluation of Definite Integrals
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;
double f (double x) {
    double a = exp(x * tan(x));
    return a;
}
int main() {
    int n, i;
    double a, b, h, sum = 0, integral;
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nEnter the limits of integration\nInitial limit a = ";    //get the limits of
integration
    cin>>a;
    cout<<"Final limit b = ";
    cin>>b;
    cout<<"Enter the no. of subintervals n = ";              // get the no. of
subintervals
    cin>>n;
    if (n%2 != 0)
    cout<<"Number of subintervals should be even";
    else {
        double x[n+1],y[n+1];
        h = (b-a)/n;                                        // get the width of the
subintervals
        for (i=0; i<=n; i++) {                              // loop to evaluate x0,...xn
and y0,...yn
            x[i] = a + i*h;
            y[i] = f(x[i]);
        }
        for (i=1; i<n; i++) {                              // loop to evaluate the sum
            if (i%2 == 0)
```

```
                sum+=2.0*y[i];
            else
                sum+=4.0*y[i];
        }
        cout<<"\nThe values of x are: ";
        for (i=0; i<=n; i++)
            cout<<setprecision(4)<<fixed<<x[i]<<" ";
        cout<<"\nThe values of y are: ";
        for (i=0; i<=n; i++)
            cout<<setprecision(4)<<fixed<<y[i]<<" ";
        cout<<endl;
        integral = (h/3.0) * (sum + y[0] + y[n]);
        cout<<"\nThe value of integral using Simpson's 1/3rd Rule is
"<<setprecision(4)<<fixed<<integral<<endl;
    }
    return 0;
}
```

OUTPUT:

For the integral $\int_{0.0}^{0.8} e^{x \tan x}\ dx$ n=8

```
FAIZAN CHOUDHARY
20BCS021

Enter the limits of integration
Initial limit a = 0.0
Final limit b = 0.8
Enter the no. of subintervals n = 8

The values of x are: 0.0000 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000
The values of y are: 1.0000 1.0101 1.0414 1.0972 1.1843 1.3141 1.5075 1.8033 2.2789

The value of integral using Simpson's 1/3rd Rule is 1.0548
```

PROGRAM 5

Weddle's Rule

CODE:

```
// Weddle's Rule for the evaluation of Definite Integrals
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;
double f (double x) {
    double a = sin(x) - log(x) + exp(x);
    return a;
}
int main() {
    int n, i;
    double a, b, h, integral =0;
```

```
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nEnter the limits of integration\nInitial limit a = ";    //get the limits of
integration
    cin>>a;
    cout<<"Final limit b = ";
    cin>>b;
    cout<<"Number of subintervals n = ";
    cin>>n;

    if (n%6 != 0)
        cout<<"\nNumber of subintervals should be a multiple of 6 for Weddle's Rule to be
applicable\n";
    else {
        int m = n/6;
        double x[n+1],y[n+1];
        h = (b-a)/n;                                    // get the width of the
subintervals
        for (i=0; i<=n; i++) {                          // loop to evaluate x0,...xn
and y0,...yn
            x[i] = a + i*h;
            y[i] = f(x[i]);
        }
        for (i=1; i<=m; i++) {                          // loop to evaluate the sum
            integral += (3.0*h/10.0) * (f(a) + f(a + 2*h) + 5*f(a+h) + 6*f(a + 3*h) + f(a
+ 4*h) + 5*f(a + 5*h)+ f(a + 6*h));
            a += 6*h;
        }
        cout<<"\nThe values of x are: ";
        for (i=0; i<=n; i++)
            cout<<setprecision(4)<<fixed<<x[i]<<" ";
        cout<<"\nThe values of y are: ";
        for (i=0; i<=n; i++)
            cout<<setprecision(4)<<fixed<<y[i]<<" ";
        cout<<endl;
        // integral = (3.0*h/10.0) * (sum + y[0] + y[n]);
        cout<<"\nThe value of integral using Weddle's Rule is
"<<setprecision(4)<<fixed<<integral<<endl;
    }
    return 0;
}
```

OUTPUT:

For the integral $\int_{0.2}^{1.4} sinx - \ln x + e^x \, dx$ n=6

```
FAIZAN CHOUDHARY
20BCS021

Enter the limits of integration
Initial limit a = 0.2
Final limit b = 1.4
Number of subintervals n = 6

The values of x are: 0.2000 0.4000 0.6000 0.8000 1.0000 1.2000 1.4000
The values of y are: 3.0295 2.7975 2.8976 3.1660 3.5598 4.0698 4.7042

The value of integral using Weddle's Rule is 4.0514
```

PROGRAM 6

Romberg Method

CODE:

```
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

#define f(x) x / sin(x)

int main() {
    double x0, xn, t[10][10], h, sm, sl, a;
    int i, k, c, r, m, p, q;

    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nEnter lower and upper limit: ";
    cin>>x0>>xn;
    cout<<"\nEnter p and q required for T(p,q): ";
    cin>>p>>q;
    h = xn - x0;
    t[0][0] = h / 2 * ((f(x0)) + (f(xn)));

    for (i=1; i<=p; i++) {
        sl = pow(2, i-1);
        sm = 0;
        for (k=1; k<=sl; k++) {
            a = x0 + (2 * k-1) * h / pow(2, i);
            sm += (f(a));
        }
        t[i][0] = t[i-1][0] / 2 + sm * h / pow(2, i);
    }
    for (c=1; c<=p; c++) {
        for (k=1; k<=c && k<=q; k++) {
            m = c - k;
            t[m+k][k] = (pow(4, k) * t[m+k][k-1] - t[m+k-1][k-1]) / (pow(4, k) - 1);
        }
    }
    cout<<"\nRomberg estimate of integration = "<<t[p][q]<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter lower and upper limit: 0 0.5

Enter p and q required for T(p,q): 2 4

Romberg estimate of integration = 1.56144e+231
```

PROGRAM 7

Numerical Double Integration

CODE:

```
// Double integration using Simpson's 1/3rd Rule
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

double f (double x, double y=1) {
    double a = sin(x*y) / (1 + x*y);
    return a;
}

int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int nx, ny, i, j;
    double a, b, c, d, h, k, z[20][20], ax[20];
    double ans;
    cout<<"\nEnter the limits of integration\nInitial limit for outer integral a = ";
    cin>>a;
    cout<<"Final limit for outer integral b = ";
    cin>>b;
    cout<<"\nInitial limit for inner integral c = ";
    cin>>c;
    cout<<"Final limit for inner integral d = ";
    cin>>d;
    cout<<"\nEnter the no. of subintervals for outer integral nx = ";
    cin>>nx;
    cout<<"Enter the no. of subintervals for inner integral ny = ";
    cin>>ny;
    h = (b-a)/nx;
    k = (d-c)/ny;

    for (i=0; i<=nx; i++) {
        for (j=0; j<=ny; j++)
            z[i][j] = f(a + i*h, c + j*k);
    }

    for (i=0; i<=nx; i++) {
        ax[i] = 0.0;
        for (j=0; j<=ny; j++) {
            if (j == 0 || j == ny)
                ax[i] += z[i][j];
            else if (j%2 == 1)
                ax[i] += 4.0*z[i][j];
            else
                ax[i] += 2.0*z[i][j];
```

```
        }
        ax[i] *= (k/3.0);
    }

    ans=0;

    for (i=0; i<=nx; i++) {
        if (i == 0 || i == nx)
            ans += ax[i];
        else if (i%2 == 1)
            ans += 4.0*ax[i];
        else
            ans += 2.0*ax[i];
    }
    ans *= (h/3.0);

    cout<<"\nThe value of the double integral with the given limits using Simpson's 1/3rd rule is "<<setprecision(4)<<fixed<<ans<<endl;

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the limits of integration
Initial limit for outer integral a = 0.0
Final limit for outer integral b = 1.0

Initial limit for inner integral c = 0.0
Final limit for inner integral d = 1.8

Enter the no. of subintervals for outer integral nx = 4
Enter the no. of subintervals for inner integral ny = 6

The value of the double integral with the given limits using Simpson's 1/3rd rule is 0.4262
```

PROGRAM 8

Gauss Elimination

CODE:

```
#include<iostream>
#include<iomanip>
using namespace std;
void display (double **a, int n) {
    int i, j;
    for (i=1; i<=n; i++) {
        for (j=1; j<=(n+1); j++)
            cout<<setprecision(2)<<fixed<<a[i][j]<<"\t";
        cout<<endl;
    }
}
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int n, i, j, k;
    cout<<"\nEnter the no. of equations: ";
    cin>>n;
    double sum = 0.0;
    double **a = new double*[n+1];
    for (i=1; i<=n; i++)
        a[i] = new double[n+2];
    double *x = new double[n+1];
    cout<<"\nEnter the elements of the augmented-matrix row-wise:\n";
    for (i=1; i<=n ;i++) {
        for (j=1; j<=(n+1) ;j++)
            cin>>a[i][j];
    }
    cout<<"\nThe entered matrix is:\n";
    display(a,n);
    for (i=1; i<=n; i++) {
        for (k=i+1; k<=n; k++) {
            double t = a[k][i] / a[i][i];
            for (j=1; j<=n+1; j++)
                a[k][j] -= t * a[i][j];
        }
    }
    x[n] = a[n][n+1] / a[n][n];
    cout<<"\n\nThe matrix after Gauss-elimination is as follows:\n";
    display(a,n);
    for (i=n-1; i>=1; i--) {
        sum = 0.0;
        x[i] = a[i][n];
        for (j=i+1; j<=n; j++)
            sum += a[i][j] * x[j];
        x[i] = (a[i][n+1] - sum) / a[i][i];
    }
    cout<<"\nThe values of the variables are as follows:\n";
```

```
    for (i=1; i<=n; i++)
        cout<<setprecision(4)<<fixed<<"\nx"<<i<<" = "<<x[i];
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the no. of equations: 3

Enter the elements of the augmented-matrix row-wise:
10 -1 2 4
1 10 -1 3
2 3 20 7

The entered matrix is:
10.00   -1.00   2.00    4.00
1.00    10.00   -1.00   3.00
2.00    3.00    20.00   7.00


The matrix after Gauss-elimination is as follows:
10.00   -1.00   2.00    4.00
-0.00   10.10   -1.20   2.60
-0.00   0.00    19.98   5.38

The values of the variables are as follows:

x1 = 0.3751
x2 = 0.2894
x3 = 0.2691
```

PROGRAM 9

Gauss Jordan

CODE:

```
#include<iostream>
#include<iomanip>
using namespace std;
void display (double **a, int n) {
    int i, j;
    for (i=1; i<=n; i++) {
        for (j=1; j<=(n+1); j++)
            cout<<setprecision(3)<<fixed<<a[i][j]<<"\t";
        cout<<endl;
    }
}
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int n, i, j, k;
    cout<<"\nEnter the no. of equations: ";
    cin>>n;
    double sum = 0.0;
    double **a = new double*[n+1];
    for (i=1; i<=n; i++)
        a[i] = new double[n+2];
    double *x = new double[n+1];
    cout<<"\nEnter the elements of the augmented-matrix row-wise:\n";
    for (i=1; i<=n ;i++) {
        for (j=1; j<=(n+1) ;j++)
            cin>>a[i][j];
    }
    cout<<"\nThe entered matrix is:\n";
    display(a,n);
    for (i=1; i<=n; i++) {
        if (a[i][i] == 0.0) {
            cout<<"Mathematical Error!";
            exit(0);
        }
        for (j=1; j<=n; j++) {
            if (i != j) {
            double t = a[j][i] / a[i][i];
            for (k=1; k<=n+1; k++)
                a[j][k] -= t * a[i][k];
            }
        }
    }
    for (i=1; i<=n; i++)
        x[i] = a[i][n+1] / a[i][i];
    cout<<"\n\nThe matrix after Gauss Jordan is as follows:\n";
    display(a,n);
    cout<<"\nThe values of the variables are as follows:\n";
```

```
    for (i=1; i<=n; i++)
        cout<<setprecision(4)<<fixed<<"\nx"<<i<<" = "<<x[i];
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the no. of equations: 4

Enter the elements of the augmented-matrix row-wise:
10 -7 3 5 6
-6 8 -1 -4 5
3 1 4 11 2
5 -9 -2 4 7

The entered matrix is:
10.000  -7.000  3.000   5.000   6.000
-6.000  8.000   -1.000  -4.000  5.000
3.000   1.000   4.000   11.000  2.000
5.000   -9.000  -2.000  4.000   7.000


The matrix after Gauss Jordan is as follows:
10.000  -0.000  -0.000  0.000   50.000
-0.000  3.800   0.000   -0.000  15.200
0.000   0.000   2.447   0.000   -17.132
-0.000  -0.000  -0.000  9.925   9.925

The values of the variables are as follows:

x1 = 5.0000
x2 = 4.0000
x3 = -7.0000
x4 = 1.0000
```

PROGRAM 10

Gauss Seidel for 4 unknowns

CODE:

```
// Program for Gauss Seidal method
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    double a[10][10], b[10], x[10], y[10];
    int n, q = 0, i = 0, j = 0;
    cout<<"\nEnter number of equations : ";
    cin>>n;
    cout<<"\nEnter coefficient matrix row-wise\n";
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++)
            cin>>a[i][j];
    }
    cout<<"\nEnter values to the right side of equation\n";
    for (i=0; i<n; i++)
        cin>>b[i];

    cout<<"\nEntered equations are:\n";
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++)
            cout<<a[i][j]<<"\t";
        cout<<":\t"<<b[i];
        cout<<endl;
    }
    cout<<"\nEnter initial values: \n";
    for (i=0; i<n; i++)
        cin>>x[i];
    cout<<"\nEnter the no. of iteration : ";
    cin>>q;
    while (q > 0) {
        for (i=0; i<n; i++) {
            y[i] = (b[i] / a[i][i]);
            for (j=0; j<n; j++) {
                if (j == i)
                    continue;
                y[i] = y[i] - ((a[i][j] / a[i][i]) * x[j]);
                x[i] = y[i];
            }
            cout<<"\nx["<<i + 1 << "] = "<<setprecision(4)<<fixed<<y[i]<<" ";
        }
        cout<<endl;
        q--;
    }
```

```
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter number of equations : 4

Enter coefficient matrix row-wise
10 -2 -1 -1
-2 10 -1 -1
-1 -1 10 -2
-1 -1 -2 10

Enter values to the right side of equation
3 15 27 -9

Entered equations are:
10      -2      -1      -1      :       3
-2      10      -1      -1      :       15
-1      -1      10      -2      :       27
-1      -1      -2      10      :       -9

Enter initial values:
0 0 0 0

Enter the no. of iteration : 5
```

```
x[1] = 0.3000
x[2] = 1.5600
x[3] = 2.8860
x[4] = -0.1368

x[1] = 0.8869
x[2] = 1.9523
x[3] = 2.9566
x[4] = -0.0248

x[1] = 0.9836
x[2] = 1.9899
x[3] = 2.9924
x[4] = -0.0042

x[1] = 0.9968
x[2] = 1.9982
x[3] = 2.9987
x[4] = -0.0008

x[1] = 0.9994
x[2] = 1.9997
x[3] = 2.9998
x[4] = -0.0001
```

PROGRAM 11

Given a set of n data points of the function y=f(x) as (x_1, y_1), (x_2, y_2), ..., (x_n, y_n). Write a program in C/C++ to fit a second degree parabola $y = a+bx+cx^2$, using Principle of least squares such that the constants a, and are to be calculated using given normal conditions:

$\Sigma y = na + b\Sigma x + c\Sigma x^2$

$\Sigma xy = a\Sigma x + b\Sigma x^2 + c\Sigma x^3$

$\Sigma x^2y = a\Sigma x^2 + b\Sigma x^3 + c\Sigma x^4$

where Σ is a summation for n number of values of x and y respectively.

CODE:

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int n, N = 2, i, j, k;
    // N is the degree of the polynomial to be fitted
    cout<<"\nEnter the number of data points: ";
    cin>>n;
    double *x = new double[n];
    double *y = new double[n];
    // to store values of sum(x^1), sum(x^2), ..., sum(x^2N)
    double *X = new double[2*N + 1];
    // to store values of sum(y), sum(xy), sum(x^2*y) ..., sum(x^N*y)
    double *Y = new double[N + 1];

    cout<<"\nEnter the data points:\n";
    for (i=0; i<n; i++) {
        cout<<"\nEnter x"<<i+1<<": ";
        cin>>x[i];
        cout<<"Enter y"<<i+1<<": ";
        cin>>y[i];
    }
    for (i=0; i<2*N+1; i++) {
        X[i] = 0;
        for (int j=0; j<n; j++)
            X[i] += pow(x[j],i);
    }
    // normal matrix to store equations
    double B[N+1][N+2];
    // to store the values of the coefficients
    double a[N+1];
    for (i=0; i<=N; i++) {
```

```
        for (j=0; j<=N; j++)
            B[i][j] = X[i+j];           // building the normal matrix by storing
corresponding coeffs at right positions except last column
    }
    for (i=0; i<N+1; i++) {
        Y[i] = 0;
        for (j=0; j<n; j++)
            Y[i] += pow(x[j],i)*y[j];
    }
    for (i=0; i<=N; i++)
        B[i][N+1] = Y[i];               // loading values of y on last column
    N=N+1;                              // for N degree we get N+1 equations
    for (i=0; i<N; i++) {
        for (k=0; k<N; k++) {
            if (B[i][i] < B[k][i]) {
                for (j=0; j<=N; j++) {
                    double temp = B[i][j];
                    B[i][j] = B[k][j];
                    B[k][j] = temp;
                }
            }
        }
    }
    for (i=0; i<N-1; i++) {
        for (k=i+1; k<N; k++) {
            double temp = B[k][i]/B[i][i];
            for (j=0; j<=N; j++)
                B[k][j] = B[k][j] - temp*B[i][j];
        }
    }
    for (i=N-1; i>=0; i--) {
        a[i] = B[i][N];                         //make the variable to be calculated equal
to the rhs of the last equation
        for (j=0; j<N; j++)
            if (j != i)
                a[i] = a[i] - B[i][j]*a[j];     //then subtract all the lhs values except
the coefficient of the variable whose value is being calculated
        a[i] = a[i]/B[i][i];                    //now finally divide the rhs by the
coefficient of the variable to be calculated
    }
    cout<<"\nThe values of the coefficients are as follows:\n";
    for (i=0; i<N; i++)
        cout<<setprecision(4)<<fixed<<"x^"<<i<<" = "<<a[i]<<endl;           // Print the
values of x^0,x^1,x^2,x^3,....
    cout<<"\nHence the fitted Polynomial is given by:\ny =";
    for (i=0; i<N; i++) {
        if (i==0)
            cout<<setprecision(4)<<fixed<<" ("<<a[i]<<")x^"<<i;
        else
            cout<<setprecision(4)<<fixed<<" + ("<<a[i]<<")x^"<<i;
    }
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of data points: 3

Enter the data points:

Enter x1: 0
Enter y1: 1

Enter x2: 1
Enter y2: 6

Enter x3: 2
Enter y3: 17

The values of the coefficients are as follows:
x^0 = 1.0000
x^1 = 2.0000
x^2 = 3.0000

Hence the fitted Polynomial is given by:
y = (1.0000)x^0 + (2.0000)x^1 + (3.0000)x^2
```

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of data points: 5

Enter the data points:

Enter x1: 0
Enter y1: 1

Enter x2: 1
Enter y2: 1.8

Enter x3: 2
Enter y3: 1.3

Enter x4: 3
Enter y4: 2.5

Enter x5: 4
Enter y5: 6.3

The values of the coefficients are as follows:
x^0 = 1.4200
x^1 = -1.0700
x^2 = 0.5500

Hence the fitted Polynomial is given by:
y = (1.4200)x^0 + (-1.0700)x^1 + (0.5500)x^2
```

PROGRAM 12

Fitting a polynomial: $y = a + bx + cx^2 + dx^3$

CODE:

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int n, N, i, j, k;
    // n is the degree of the polynomial to be fitted
    cout<<"\nEnter the number of data points: ";
    cin>>N;
    double x[N], y[N];
    cout<<"\nEnter the data points:\n";
    cout<<"\nEnter x values:\n";
    for (i=0; i<N; i++)
        cin>>x[i];
    cout<<"\nEnter y values:\n";
    for (i=0; i<N; i++)
        cin>>y[i];
    cout<<"\nEnter the degree of the polynomial: ";
    cin>>n;
    // to store values of sum(x^1), sum(x^2), ..., sum(x^2N)
    double X[2*n+1];
    for (i=0; i<2*n+1; i++) {
        X[i] = 0;
        for (j=0; j<N; j++)
            X[i] += pow(x[j],i);
    }
    // normal matrix to store equations
    double B[n+1][n+2];
    // to store the values of the coefficients
    double a[n+1];
    for (i=0; i<=n; i++) {
        for (j=0; j<=n; j++)
            B[i][j] = X[i+j];           // building the normal matrix by storing
corresponding coeffs at right positions except last column
    }
    // to store values of sum(y), sum(xy), sum(x^2*y) ..., sum(x^N*y)
    double Y[n + 1];
    for (i=0; i<n+1; i++) {
        Y[i] = 0;
        for (j=0; j<N; j++)
            Y[i] += pow(x[j],i)*y[j];
    }
    for (i=0; i<=n; i++)
        B[i][n+1] = Y[i];               // loading values of y on last column
    n=n+1;                              // for N degree we get N+1 equations
```

```
    for (i=0; i<n; i++) {
        for (k=i+1; k<n; k++) {
            if (B[i][i] < B[k][i]) {
                for (j=0; j<=n; j++) {
                    double temp = B[i][j];
                    B[i][j] = B[k][j];
                    B[k][j] = temp;
                }
            }
        }
    }
    for (i=0; i<n-1; i++) {
        for (k=i+1; k<n; k++) {
            double temp = B[k][i]/B[i][i];
            for (j=0; j<=n; j++)
                B[k][j] = B[k][j] - temp*B[i][j];
        }
    }
    for (i=n-1; i>=0; i--) {
        a[i] = B[i][n];                         //make the variable to be calculated equal
to the rhs of the last equation
        for (j=0; j<n; j++)
            if (j != i)
                a[i] = a[i] - B[i][j]*a[j];     //then subtract all the lhs values except
the coefficient of the variable whose value is being calculated
        a[i] = a[i]/B[i][i];                    //now finally divide the rhs by the
coefficient of the variable to be calculated
    }
    cout<<"\nThe values of the coefficients are as follows:\n";
    for (i=0; i<n; i++)
        cout<<setprecision(4)<<fixed<<"x^"<<i<<" = "<<a[i]<<endl;            // Print the
values of x^0,x^1,x^2,x^3,....
    cout<<"\nHence the fitted Polynomial is given by:\ny =";
    for (i=0; i<n; i++) {
        if (i==0)
            cout<<setprecision(4)<<fixed<<" ("<<a[i]<<")x^"<<i;
        else
            cout<<setprecision(4)<<fixed<<" + ("<<a[i]<<")x^"<<i;
    }
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of data points: 5

Enter the data points:

Enter x values:
1 5 7 9 12

Enter y values:
10 15 12 15 21

Enter the degree of the polynomial: 3
```

```
The values of the coefficients are as follows:
x^0 = 6.4641
x^1 = 4.3637
x^2 = -0.7817
x^3 = 0.0433

Hence the fitted Polynomial is given by:
y = (6.4641)x^0 + (4.3637)x^1 + (-0.7817)x^2 + (0.0433)x^3
```

PROGRAM 13

Bisection Method

CODE:

```
/*Program for bisection method*/
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

double func (double x) {          // Function definition
    return cos(x) - x*exp(x);
}

int main() {
    double a, b, c, e;
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nEnter initial approximations of the root: ";
    cin>>a>>b;
    cout<<"\nEnter the value of the tolerance: ";
    cin>>e;

    if (func(a) * func(b) >= 0)
        cout<<"\nRoot is not lying between a and b\n";
    else {
        for(int i=1; i<=15; i++) {
            c = (a+b) / 2.0;
            if (func(c) == 0)
                break;
            else {
                if (abs(b - a) <= e)
                    break;
                if (func(a) * func(c) < 0)
                    b = c;
                else
                    a = c;
            }
            cout<<"\nIteration: "<<i<<"\tApproximation to the root is:
"<<setprecision(4)<<fixed<<c<<endl;
        }
    }
    cout<<"\nTherefore the root of the given function is:
"<<setprecision(4)<<fixed<<c<<endl;
    return 0;
}
```

OUTPUT:

For function f(x) = cos(x) - xe^x

```
FAIZAN CHOUDHARY
20BCS021

Enter initial approximations of the root: 0 1

Enter the value of the tolerance: 0.0001

Iteration: 1    Approximation to the root is: 0.5000

Iteration: 2    Approximation to the root is: 0.7500

Iteration: 3    Approximation to the root is: 0.6250

Iteration: 4    Approximation to the root is: 0.5625

Iteration: 5    Approximation to the root is: 0.5312

Iteration: 6    Approximation to the root is: 0.5156

Iteration: 7    Approximation to the root is: 0.5234

Iteration: 8    Approximation to the root is: 0.5195

Iteration: 9    Approximation to the root is: 0.5176

Iteration: 10   Approximation to the root is: 0.5186

Iteration: 11   Approximation to the root is: 0.5181

Iteration: 12   Approximation to the root is: 0.5178

Iteration: 13   Approximation to the root is: 0.5177

Iteration: 14   Approximation to the root is: 0.5178

Therefore the root of the given function is: 0.5177
```

PROGRAM 14

Newton Raphson Method

CODE:

```
/*Program for Newton Raphson method*/
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

double f (double x) {
    return cos(x) - x*exp(x);
}

double df (double x) {        // Derivative of Function
    return -sin(x) - exp(x) - x*exp(x);
}

int main() {
    double a, b, e;
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nEnter initial approximation of the root: ";
    cin>>a;
    cout<<"\nEnter the value of the tolerance: ";
    cin>>e;

    for (int i=1; i<=15; i++) {
        b = a - (f(a) / df(a));
        if (f(b) == 0) {
            cout<<"\nActual root is: "<<b;
            break;
        }
        else {
            if (abs(b - a) <= e)
                break;
            else
                a = b;
            cout<<"\nIteration: "<<i<<"\tApproximation to the root is:
"<<setprecision(4)<<fixed<<b<<endl;
        }
    }
    cout<<"\nTherefore the root of the given function is:
"<<setprecision(4)<<fixed<<b<<endl;
    return 0;
}
```

OUTPUT:

For function f(x) = cos(x) - xe^x

```
FAIZAN CHOUDHARY
20BCS021

Enter initial approximation of the root: 0

Enter the value of the tolerance: 0.0001

Iteration: 1    Approximation to the root is: 1.0000

Iteration: 2    Approximation to the root is: 0.6531

Iteration: 3    Approximation to the root is: 0.5313

Iteration: 4    Approximation to the root is: 0.5179

Iteration: 5    Approximation to the root is: 0.5178

Therefore the root of the given function is: 0.5178
```

PROGRAM 15

Regula-Falsi Method

CODE:

```cpp
/*Program for Regula Falsi method*/
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;
double f (double x) {
    return x*log10(x) - 1.2;
}
int main() {
    double a, b, c, e;
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nEnter initial approximation of the root: ";
    cin>>a>>b;
    cout<<"\nEnter the value of the tolerance: ";
    cin>>e;
    if (f(a) * f(b) >= 0)
        cout<<"\nRoot is not lying between a and b\n";
    else {
        for(int i=1; i<=15; i++) {
            c = (a+b) / 2.0;
            if (f(c) == 0)
                break;
            else {
                if (abs(b - a) <= e)
```

```
                    break;
                if (f(a) * f(c) < 0)
                    b = c;
                else
                    a = c;
            }
            cout<<"\nIteration: "<<i<<"\tApproximation to the root is:
"<<setprecision(4)<<fixed<<c<<endl;
        }
    }

    cout<<"\nTherefore the root of the given function is:
"<<setprecision(4)<<fixed<<b<<endl;
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter initial approximation of the root: 2 3

Enter the value of the tolerance: 0.0001

Iteration: 1    Approximation to the root is: 2.5000

Iteration: 2    Approximation to the root is: 2.7500

Iteration: 3    Approximation to the root is: 2.6250

Iteration: 4    Approximation to the root is: 2.6875

Iteration: 5    Approximation to the root is: 2.7188

Iteration: 6    Approximation to the root is: 2.7344

Iteration: 7    Approximation to the root is: 2.7422

Iteration: 8    Approximation to the root is: 2.7383

Iteration: 9    Approximation to the root is: 2.7402

Iteration: 10   Approximation to the root is: 2.7412

Iteration: 11   Approximation to the root is: 2.7407

Iteration: 12   Approximation to the root is: 2.7405

Iteration: 13   Approximation to the root is: 2.7406

Iteration: 14   Approximation to the root is: 2.7407

Therefore the root of the given function is: 2.7407
```

PROGRAM 16

Write a program to generate first 100 prime numbers leaving those divisible by 5 and 7 [Prime Number Problem]

CODE:

```
#include <iostream>
#include <math.h>
using namespace std;
int isPrime (int n) {
    int prime = 1;
    int j;
    for (j=2; j<=sqrt(n); j++) {
        if (n%j==0) {
            prime = 0;
            break;
        }
    }
    return prime;
}
int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int counter=0;
    cout<<"\nFirst 100 prime numbers generated not divisible by 5 and 7 are:\n";
    for (int i=2; counter<101; i++) {
        if (isPrime(i)) {
            if (i%5 !=0 && i%7 !=0) {
                if (counter == 30 || counter == 50 || counter == 70 || counter == 90)
                    cout<<endl;
                cout<<i<<" ";
            }
            counter++;
        }
    }
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

First 100 prime numbers generated not divisible by 5 and 7 are:
2 3 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541 547
```

PROGRAM 17

Runge-Kutta method for system of differential equations

CODE:

```
// to solve system of differential equations using Runge-Kutta method.
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

double f1 (double x, double y, double z) {
    return (1 + x*z);
}

double f2 (double x, double y, double z) {
    return (-x*y);
}

int main () {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    double k1, k2, k3, k4, l1, l2, l3, l4;
    double x0, y0, z0, h, x, n;

    cout<<"\nEnter initial values of x, y, z: ";
    cin>>x0>>y0>>z0;
    cout<<"\nEnter the value of the step size: ";
    cin>>h;
    cout<<"\nEnter the value of x where the solution is to be found: ";
    cin>>x;
    n = (x - x0) / h;

    for (int i=1; i<=n; i++) {
        k1 = h * f1(x0, y0, z0);
        l1 = h * f2(x0, y0, z0);
        k2 = h * f1(x0 + 0.5 * h, y0 + 0.5 * k1, z0 + 0.5 * l1);
        l2 = h * f2(x0 + 0.5 * h, y0 + 0.5 * k1, z0 + 0.5 * l1);
        k3 = h * f1(x0 + 0.5 * h, y0 + 0.5 * k2, z0 + 0.5 * l2);
        l3 = h * f2(x0 + 0.5 * h, y0 + 0.5 * k2, z0 + 0.5 * l2);
        k4 = h * f1(x0 + h, y0 + k3, z0 + l3);
        l4 = h * f2(x0 + h, y0 + k3, z0 + l3);
        x0 = x0 + h;
        y0 = y0 + (1.0/6.0) * (k1 + 2 * k2 + 2 * k3 + k4);
        z0 = z0 + (1.0/6.0) * (l1 + 2 * l2 + 2 * l3 + l4);

        cout<<"\nAfter iteration "<<i<<" the values of x, y, z are:
"<<setprecision(4)<<fixed<<x0<<" "<<y0<<" "<<z0<<endl;
    }
    return 0;
}
```

OUTPUT:

For $\frac{dy}{dx} = 1 + xz$, $\frac{dz}{dx} = -xy$ y(0) = 0.0, z(0) = 1.0, at x = 1.0 and h = 0.2

```
FAIZAN CHOUDHARY
20BCS021

Enter initial values of x, y, z: 0 0 1

Enter the value of the step size: 0.2

Enter the value of x where the solution is to be found: 1.0

After iteration 1 the values of x, y, z are: 0.2000 0.2200 0.9971

After iteration 2 the values of x, y, z are: 0.4000 0.4792 0.9755

After iteration 3 the values of x, y, z are: 0.6000 0.7739 0.9121

After iteration 4 the values of x, y, z are: 0.8000 1.0929 0.7806

After iteration 5 the values of x, y, z are: 1.0000 1.4138 0.5537
```

PROGRAM 18

Milne's Predictor-Corrector Method

CODE:

```cpp
// Milne's Method
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

double f (double a, double b) {
    double r = (a*a + 3*b*b) / 10;
    return r;
}

int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    double x0, y0, h, xn, x[10], y[10];
    double f1, f2, f3, f4, f5, k1, k2, k3, k4, yi, yc;
    int i, n;

    cout<<"\nEnter the initial values of x, y: ";
```

```
    cin>>x0>>y0;
    cout<<"\nEnter the value of the step size: ";
    cin>>h;
    cout<<"\nEnter the final value of x: ";
    cin>>xn;
    n = (xn - x0) / h;
    x[0] = x0;
    y[0] = y0;

    for (i=1; i<=n; i++)
        x[i] = x[0] + i*h;

    for (i=1; i<=n; i++) {
        k1 = h * f(x0, y0);
        k2 = h * f(x0 + 0.5 * h, y0 + 0.5 * k1);
        k3 = h * f(x0 + 0.5 * h, y0 + 0.5 * k2);
        k4 = h * f(x0 + h, y0 + k3);
        y[i] = y[i-1] + (1.0/6.0) * (k1 + 2 * k2 + 2 * k3 + k4);
        x0 = x0 + h;
        y0 = y[i];
    }

    cout<<"\nThe values of x and y are: "<<endl;
    cout<<"x\ty"<<endl;
    for (i=0; i<=n; i++)
        cout<<x[i]<<"\t"<<y[i]<<endl;

    f1 = f(x[1], y[1]);
    f2 = f(x[2], y[2]);
    f3 = f(x[3], y[3]);

    // Milne's Predictor
    y[4] = y[0] + (4*h/3.0) * (2*f1 - f2 + 2*f3);
    A:
    // cout<<"Hello";
    f4 = f(x[4], y[4]);

    cout<<"\nThe predicted value of y[4] is: "<<y[4]<<endl;
    // Milne's Corrector
    yi = y[2] + (h/3.0) * (f2 + 4*f3 + f4);

    if (fabs((yi - y[4]) / yi) <= 0.0001)
        goto B;
    else {
        y[4] = yi;
        goto A;
    }

    B:
    y[5] = y[1] + (4*h/3.0) * (2*f2 - f3 + 2*f4);
    f5 = f(x[5], y[5]);
    yc = y[3] + (h/3.0) * (f3 + 2*f4 + f5);
    while (fabs((yc - y[5]) / yc) <= 0.0001) {
        y[5] = yc;
```

```
        f5 = f(x[5], y[5]);
        yc = y[3] + (h/3.0) * (f3 + 2*f4 + f5);
    }


    cout<<"\nThe corrected values of x, y are: "<<endl;
    cout<<"x\ty"<<endl;
    for (i=0; i<=n; i++)
        cout<<x[i]<<"\t"<<y[i]<<endl;

    return 0;
}
```

OUTPUT:

For $10\frac{dy}{dx} = x^2 + 3y^2$ y(1) = 2 1<x<=2.0, h=0.2

```
FAIZAN CHOUDHARY
20BCS021

Enter the initial values of x, y: 1 2

Enter the value of the step size: 0.2

Enter the final value of x: 2

The values of x and y are:
x       y
1       2
1.2     2.30025
1.4     2.70785
1.6     3.28712
1.8     4.16734

The predicted value of y[4] is: 4.14988

The predicted value of y[4] is: 4.16628

The predicted value of y[4] is: 4.169

The predicted value of y[4] is: 4.16946

The corrected values of x, y are:
x       y
1       2
1.2     2.30025
1.4     2.70785
1.6     3.28712
1.8     4.16946
```

PROGRAM 19

Write a program in C/C++ to print the sum and average of marks obtained by 60 students in 10 different subjects. Also write the program segment to find the marks of the student ranking first in class.

CODE:

```
#include <iostream>
#include <limits.h>
using namespace std;

// reduced number of variables for easier input
const int MAX = 3, NUM = 3;

struct student {
    int id;
    char name[20];
    float marks[NUM];
    float total;
    float avg;
} st[MAX];

void input () {
    for (int i=0; i<MAX; i++) {
        st[i].id = i+1;
        st[i].total = 0;
        st[i].avg = 0;
        cout<<"\nEnter the details of student "<<i+1<<": ";
        // cout<<"\nEnter the ID: ";
        // cin>>st[i].id;
        cout<<"\nEnter the name: ";
        fflush(stdin);
        cin.getline(st[i].name, 20);
        cout<<"\nEnter the marks in three subjects: ";
        for (int j=0; j<NUM; j++) {
            cin>>st[i].marks[j];
            st[i].total += st[i].marks[j];
        }
        st[i].avg = st[i].total / NUM;
    }
}

void display () {
    cout<<"\nID\tName\t\t\tMarks\t\tTotal\tAverage\n";
    for (int i=0; i<MAX; i++) {
        cout<<st[i].id<<"\t"<<st[i].name<<"\t";
        for (int j=0; j<NUM; j++) {
            cout<<st[i].marks[j]<<"\t";
        }
        cout<<st[i].total<<"\t"<<st[i].avg<<endl;
```

```
    }
}

int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int i, idx;
    float sum = 0, max = INT_MIN;

    input();
    display();
    for (i=0; i<MAX; i++) {
        if (st[i].total > max) {
            max = st[i].total;
            idx = i;
        }
    }

    cout<<"\nThe student with the highest total is: "<<endl;
    cout<<st[idx].id<<"\t"<<st[idx].name<<"\t";
    for (int j=0; j<NUM; j++) {
        cout<<st[idx].marks[j]<<"\t";
    }
    cout<<st[idx].total<<"\t"<<st[idx].avg<<endl;

    return 0;
}
```

OUTPUT: (reduced input values)

```
FAIZAN CHOUDHARY
20BCS021

Enter the details of student 1:
Enter the name: Kiran Rijiju

Enter the marks in three subjects: 69 98 90

Enter the details of student 2:
Enter the name: Usman Khan

Enter the marks in three subjects: 95 93 87

Enter the details of student 3:
Enter the name: Tushar Kumar

Enter the marks in three subjects: 78 92 96

ID      Name                    Marks           Total   Average
1       Kiran Rijiju    69      98      90      257     85.6667
2       Usman Khan      95      93      87      275     91.6667
3       Tushar Kumar    78      92      96      266     88.6667

The student with the highest total is:
2       Usman Khan      95      93      87      275     91.6667
```

PROGRAM 20

Let A and B be any two matrices of order l×m and m×n respectively. Write a program in C/C++ to find a matrix C such that C = A×B using function. Also write its main program CALLING the function.

CODE:

```
#include <iostream>
using namespace std;
int l, m, n, p;
int **C = new int*[l];

void multiply (int **A, int **B, int l, int m, int n) {
    for (int i=0; i<l; i++) {                   //for accessing the rows of A
        for (int j=0; j<n; j++) {               //for accessing the columns of B
            C[i][j]=0;
            for (int k=0; k<m; k++)             //to traverse and multiply
                C[i][j]+=A[i][k]*B[k][j];
        }
    }
}

void display (int **A, int l, int m) {
    for (int i=0; i<l; i++) {
        for (int j=0; j<m; j++)
            cout<<A[i][j]<<" ";
        cout<<endl;
    }
}

int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    cout<<"\nEnter the number of rows and columns for matrix A: ";
    cin>>l>>m;
    cout<<"\nEnter the number of rows and columns for matrix B: ";
    cin>>p>>n;
    if (p != m) {
        cout<<"\nMatrix not compatible for multiplication! ";
        return 0;
    }
    int **A = new int*[l];
    int **B = new int*[m];
    for (int i=0; i<l; i++) {
        A[i] = new int[m];
        C[i] = new int[n];
    }
    for (int i=0; i<m; i++)
        B[i] = new int[n];
```

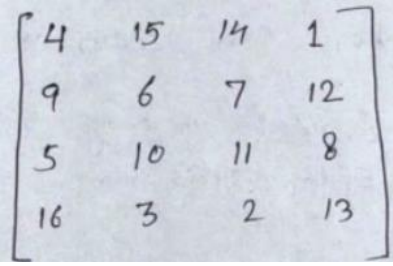
```
    cout<<"\nEnter the elements of matrix A row-wise:\n";
    for (int i=0; i<l; i++) {
        for (int j=0; j<m; j++)
            cin>>A[i][j];
    }

    cout<<"\nEnter the elements of matrix B row-wise:\n";
    for (int i=0; i<m; i++) {
        for (int j=0; j<n; j++)
            cin>>B[i][j];
    }
    cout<<"\nInput matrix A:\n";
    display(A, l, m);
    cout<<"\nInput matrix B:\n";
    display(B, m, n);

    multiply (A, B, l, m, n);
    cout<<"\nOutput matrix C (after multiplication):\n";
    display(C, l, n);
    cout<<endl;

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of rows and columns for matrix A: 2 2

Enter the number of rows and columns for matrix B: 2 3

Enter the elements of matrix A row-wise:
1 2 3 4

Enter the elements of matrix B row-wise:
5 6 7 8 9 10

Input matrix A:
1 2
3 4

Input matrix B:
5 6 7
8 9 10

Output matrix C (after multiplication):
21 24 27
47 54 61
```

PROGRAM 21

Magic square problem A magic square is a square matrix of integers such that the sum of every row, the sum of every column, and sum of each of the diagonals are equal, such a magic square given below. Write a program using C/C++ to read the elements of the square matrix and check whether the matrix represents a square matrix



A magic square with sum = 34

CODE:

```
#include <iostream>
#include <algorithm>
using namespace std;
int sum;

bool areDuplicates (int **arr, int N) {
    int i, j;
    int *b = new int[N*N];
    for (i=0; i<N; i++) {
        for (j=0; j<N; j++)
            b[i*N+j] = arr[i][j];
    }
    sort(b, b+N*N);
    for (i=0; i<N*N-1; i++)
        if (b[i] == b[i+1] && b[i] > 0)
            return true;
    return false;
}

int rowSum (int **arr, int N) {
    int i, j;
    long sum_row = 0, prev_sum_row = 0;
    for (i=0; i<N; i++) {
        for (j=0; j<N; j++)
            sum_row += arr[i][j];
        if (i!= 0 && sum_row != prev_sum_row)
            return 0;
        prev_sum_row = sum_row;
        sum_row = 0;
    }
    cout<<"\nThe sum of the rows is: "<<prev_sum_row;
```

```
    return prev_sum_row;
}

int colSum (int **arr, int N) {
    int i, j;
    long sum_col = 0, prev_sum_col = 0;
    for (i=0; i<N; i++) {
        for (j=0; j<N; j++)
            sum_col += arr[j][i];
        if (i!= 0 && sum_col != prev_sum_col)
            return 0;
        prev_sum_col = sum_col;
        sum_col = 0;
    }
    cout<<"\nThe sum of the columns is: "<<prev_sum_col;
    return prev_sum_col;
}

int diagSum (int **arr, int N) {
    int i, j;
    long diag1 = 0, diag2 = 0;
    for (i=0; i<N; i++) {
        for (j=0; j<N; j++) {
            if (i == j)
                diag1 += arr[i][j];
            if (i + j == N-1)
                diag2 += arr[i][j];
        }
    }
    if (diag1 != diag2)
        return 0;
    cout<<"\nThe sum of the diagonals is: "<<diag1<<endl;
    return diag1;
}

int main() {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int N;
    cout<<"\nEnter the order of the matrix: ";
    cin>>N;

    int **A = new int*[N];
    for (int i=0; i<N; i++) {
        A[i] = new int[N];
    }

    cout<<"\nEnter the elements of the matrix:\n";
    for (int i=0; i<N; i++) {
        for (int j=0; j<N; j++) {
            cin>>A[i][j];
        }
    }
```

```
    // magic square condition check: no duplicates
    if (areDuplicates(A, N)) {
        cout<<"\nThe matrix cannot be a magic square.";
        return 0;
    }

    // magic square condition check: sum of rows and columns and diagonals are same
    sum = N*(N*N+1)/2;

    if (rowSum(A, N) == 0 || colSum(A, N) == 0 || diagSum(A, N) == 0) {
        cout<<"\nThe matrix is not a magic square.";
        return 0;
    }
    cout<<"\nThe matrix is a magic square.";

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the order of the matrix: 3

Enter the elements of the matrix:
2 9 4
7 5 3
6 1 8

The sum of the rows is: 15
The sum of the columns is: 15
The sum of the diagonals is: 15

The matrix is a magic square.
```

```
FAIZAN CHOUDHARY
20BCS021

Enter the order of the matrix: 4

Enter the elements of the matrix:
4 15 14 1
9 6 7 12
5 10 11 8
16 3 2 13

The sum of the rows is: 34
The sum of the columns is: 34
The sum of the diagonals is: 34

The matrix is a magic square.
```