```
/*********************************************************
*                        1                               *
*********************************************************/
#include <bits/stdc++.h>
using namespace std;

int main()
{
    double allX[] = {1, 2, 3, 4, 5};
    double allY[] = {1, 8, 27, 64, 125};
    int tableSize = sizeof allY / sizeof allY[0];
    vector<double>delY[tableSize];
    double x = 1;
    double h = allX[1] - allX[0];
    double u = (x - allX[0]) / h;

    for (int i = 0; i < tableSize; i++) {
        delY[0].push_back(allY[i]);
    }

    for (int i = 1; i < tableSize; i++) {
        for (int j = 1; j < delY[i - 1].size(); j++) {
            delY[i].push_back(delY[i - 1][j] - delY[i - 1][j - 1]);
        }
    }

    double firstD = delY[1][0] + (2 * u - 1) * (delY[2][0] / 2) + (3
* u * u - 6 * u + 2) * (delY[3][0] / 6) + (4 * u * u * u - 18 * u * u + 22
* u - 6) * (delY[4][0] / 24);
    firstD /= h;

    double secondD = delY[2][0] + (6 * u - 6) * (delY[3][0] / 6) + (
12 * u * u - 36 * u + 22) * (delY[4][0] / 24);
    secondD /= (h * h);

    cout << "First Derivative = " << setprecision(3) << fixed <<
firstD << "\n";
    cout << "Second Derivative = " << setprecision(3) << fixed <<
secondD << "\n";

    return 0;
}

/*********************************************************
*                        2                               *
*********************************************************/
#include <bits/stdc++.h>
using namespace std;

int main()
{
    double allX[] = {1, 2, 3, 4, 5};
    double allY[] = {1, 8, 27, 64, 125};
    int tableSize = sizeof allY / sizeof allY[0];
    vector<double>delY[tableSize];
    double x = 1.5;
    double h = allX[1] - allX[0];
    double u = (x - allX[0]) / h;

    for (int i = 0; i < tableSize; i++) {
        delY[0].push_back(allY[i]);
    }

    for (int i = 1; i < tableSize; i++) {
        for (int j = 1; j < delY[i - 1].size(); j++) {
            delY[i].push_back(delY[i - 1][j] - delY[i - 1][j - 1]);
```

```cpp
        }
    }

    double firstD = delY[1][0] + (2 * u - 1) * (delY[2][0] / 2) + (3 * u * u - 6 * u + 2) * (delY[3][0] / 6) + (4 * u * u * u - 18 * u * u + 22 * u - 6) * (delY[4][0] / 24);
    firstD /= h;

    double secondD = delY[2][0] + (6 * u - 6) * (delY[3][0] / 6) + (12 * u * u - 36 * u + 22) * (delY[4][0] / 24);
    secondD /= (h * h);

    cout << "First Derivative = " << setprecision(3) << fixed << firstD << "\n";
    cout << "Second Derivative = " << setprecision(3) << fixed << secondD << "\n";

    return 0;
}

/********************************************************
*                            3                          *
********************************************************/
#include <bits/stdc++.h>
using namespace std;

double fX(double x)
{
    return (5 * log10(x));
}

int main()
{
    double upperLimit = 7;
    double lowerLimit = 1;
    int n = 6;
    double h = (upperLimit - lowerLimit) / n;

    double x = lowerLimit;
    double y[n + 1];
    for (int i = 0; i <= n; i++) {
        y[i] = fX(x);
        x += h;
    }

    double I = y[0] + y[n];
    for (int i = 1; i < n; i++) {
        I += (2 * y[i]);
    }
    I = I * (h / 2);

    cout << "Area = " << setprecision(3) << fixed << I << "\n";

    return 0;
}

/********************************************************
*                            4                          *
********************************************************/
#include <bits/stdc++.h>
using namespace std;

#define PI acos(-1.0)
#define sinD(degree) sin((degree * PI) / 180.0)

double fX(double x)
```

```cpp
{
    return (PI * exp(sinD(x))) / 2;
}

int main()
{
    double upperLimit = 7;
    double lowerLimit = 1;
    int n = 6;
    double h = (upperLimit - lowerLimit) / n;

    double x = lowerLimit;
    double y[n + 1];
    for (int i = 0; i <= n; i++) {
        y[i] = fX(x);
        x += h;
    }

    double I = y[0] + y[n];
    for (int i = 1; i < n; i++) {
        if (i % 2 == 0) {
            I += (2 * y[i]);
        }
        else {
            I += (4 * y[i]);
        }
    }
    I = I * (h / 3);

    cout << "Area = " << setprecision(3) << fixed << I << "\n";

    return 0;
}

/********************************************************
*                       5                               *
********************************************************/
#include <bits/stdc++.h>
using namespace std;

double fX(double x)
{
    return (x / (1 + x * x));
}

int main()
{
    double upperLimit = 7;
    double lowerLimit = 1;
    int n = 6;
    double h = (upperLimit - lowerLimit) / n;

    double x = lowerLimit;
    double y[n + 1];
    for (int i = 0; i <= n; i++) {
        y[i] = fX(x);
        x += h;
    }

    double I = y[0] + y[n];
    for (int i = 1; i < n; i++) {
        if (i % 3 == 0) {
            I += (2 * y[i]);
        }
        else {
            I += (3 * y[i]);
        }
    }
    I = I * ((3 * h) / 8);

    cout << "Area = " << setprecision(3) << fixed << I << "\n";

    return 0;
}

/**********************************************************
*                          6                              *
**********************************************************/
#include <bits/stdc++.h>
using namespace std;

int temp[4][4];

int det3x3()
{
    int ans = 0;
    ans += temp[1][1] * (temp[2][2] * temp[3][3] - temp[2][3] * temp[3][2]);
    ans -= temp[1][2] * (temp[2][1] * temp[3][3] - temp[2][3] * temp[3][1]);
    ans += temp[1][3] * (temp[2][1] * temp[3][2] - temp[2][2] * temp[3][1]);
    return ans;
}

int main()
{
    int n = 4;
    int matrix[n + 1][n + 1];

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin >> matrix[i][j];
        }
    }

    int ans = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 2; j <= n; j++) {
            int m = 1;
            for (int k = 1; k <= n; k++) {
                if (k == i) {
                    continue;
                }
                temp[j - 1][m] = matrix[j][k];
                m++;
            }
        }

        cout << det3x3() << "\n";

        int d = matrix[1][i] * det3x3();
        if (i % 2 == 1) {
            ans += d;
        }
        else {
            ans -= d;
        }
    }

    cout << "Determinant = " << ans << "\n";

```

```cpp
    return 0;
}

/********************************************************
*                       7                               *
********************************************************/
#include <bits/stdc++.h>
using namespace std;

const int n = 3;

double A[n + 1][n + 1] = {
                {0, 0, 0, 0},
                {0, 1, 1, 1},
                {0, 1, 2, 3},
                {0, 1, 3, 4}
                };

double B[n + 1] = {0, 1, 6, 6};

double det3x3()
{
    double ans = 0;
    ans += A[1][1] * (A[2][2] * A[3][3] - A[2][3] * A[3][2]);
    ans -= A[1][2] * (A[2][1] * A[3][3] - A[2][3] * A[3][1]);
    ans += A[1][3] * (A[2][1] * A[3][2] - A[2][2] * A[3][1]);
    return ans;
}

int main()
{
    double detA = det3x3();
    double cofactor[n + 1][n + 1];

    vector<double>data;

    data.clear();
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            for (int row = 1; row <= n; row++) {
                for (int col = 1; col <= n; col++) {
                    if (row == i || col == j) {
                        continue;
                    }
                    data.push_back(A[row][col]);
                }
            }

            cofactor[i][j] = (data[0] * data[3]) - (data[1] * data[2]);
            if ((i + j) % 2 == 1) {
                cofactor[i][j] *= -1;
            }
            data.clear();
        }
    }

    double adjA[n + 1][n + 1];
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            adjA[i][j] = cofactor[j][i];
            adjA[i][j] /= detA;
        }
    }

    double solution[n + 1];
    for (int i = 1; i <= n; i++) {
        double x = 0;
        for (int j = 1; j <= n; j++) {
            x += adjA[i][j] * B[j];
        }
        solution[i] = x;
    }

    cout << "x = " << setprecision(3) << fixed << solution[1] << "\n";
    cout << "y = " << setprecision(3) << fixed << solution[2] << "\n";
    cout << "z = " << setprecision(3) << fixed << solution[3] << "\n";

    return 0;
}

/*********************************************************
*                        11                              *
*********************************************************/
#include <bits/stdc++.h>
using namespace std;

double eqX(double y, double z)
{
    double x = (95 - 11 * y + 4 * z) / 83;
    return x;
}

double eqY(double x, double z)
{
    double y = (71 - 3 * x - 29 * z) / 8;
    return y;
}

double eqZ(double x, double y)
{
    double z = (104 - 7 * x - 52 * y) / 13;
    return z;
}

int main()
{
    double x, y, z;
    int n = 5;

    x = y = z = 0;

    for (int i = 1; i <= n; i++) {
        double newX = eqX(y, z);
        double newY = eqY(x, z);
        double newZ = eqZ(x, y);

        x = newX;
        y = newY;
        z = newZ;
    }

    cout << "x = " << setprecision(3) << fixed << x << "\n";
    cout << "y = " << setprecision(3) << fixed << y << "\n";
    cout << "z = " << setprecision(3) << fixed << z << "\n";

    return 0;
}
```

```
/*********************************************************
*                        12                              *
*********************************************************/
#include <bits/stdc++.h>
using namespace std;

double eqX(double y, double z)
{
    double x = (12 - y - z) / 10;
    return x;
}

double eqY(double x, double z)
{
    double y = (13 - 2 * x - z) / 10;
    return y;
}

double eqZ(double x, double y)
{
    double z = (14 - 2 * x - 2 * y) / 10;
    return z;
}

int main()
{
    double x, y, z;
    int n = 1000;

    x = y = z = 0;

    for (int i = 1; i <= n; i++) {
        x = eqX(y, z);
        y = eqY(x, z);
        z = eqZ(x, y);
    }

    cout << "x = " << setprecision(3) << fixed << x << "\n";
    cout << "y = " << setprecision(3) << fixed << y << "\n";
    cout << "z = " << setprecision(3) << fixed << z << "\n";

    return 0;
}
```