

Numerical Analysis

🕒 Created	@March 17, 2023 8:55 PM
🕒 Last Edited Time	@March 19, 2023 6:04 PM
☰ By	Sagor <ASH2101008M>
@ Email	

```
//e^(-3x/2)

#include<stdio.h>
#include<math.h>

int main(){

    double ans=0;
    for(double i=0.1; i<=2; i+=0.1){
        ans += exp((-1)*i*i/2);
    }

    printf("%lf", ans);
}
```

```
//5x^3 + e^(-2x)

#include<stdio.h>
#include<math.h>

int main(){

    double ans=0;
    for(double i=0.1; i<=2; i+=0.1){
        ans += 5*i*i*i + exp((-1)*2*i);
    }

    printf("%lf", ans);
}
```

X Maclaurin Series

$$e^x = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \dots$$

$$\ln(1+x) = \frac{x}{1!} - \frac{x^2}{2!} + \frac{x^3}{3!} - \dots$$

$$\ln(1-x) = - \left(x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \right)$$

```
//e^x
#include<stdio.h>
#include<math.h>

int main(){

    double ans=1;
    double x;
    scanf("%lf", &x);

    double factorial=1;

    for(double i=2; i<=6; i++){
        ans += pow(x, i-1)/factorial;
        factorial*=i;
    }

    printf("%lf", ans);
}
```

```
//ln(1+x)
#include<stdio.h>
#include<math.h>

int main(){

    double ans=0;
    double x;
    scanf("%lf", &x);

    double factorial=1;
```

```

for(int i=1; i<=6; i++){
    if(i%2==0)
        ans -= pow(x, i)/factorial;
    else
        ans += pow(x, i)/factorial;

    factorial*=i;
}

printf("%lf", ans);
}

```

```

// ln (1-x)
#include<stdio.h>
#include<math.h>

int main(){

    double ans=0;
    double x;
    scanf("%lf", &x);

    double factorial=1;

    for(int i=1; i<=6; i++){
        ans += pow(x, i)/factorial;
        factorial*=i;
    }
    ans *= (-1);
    printf("%lf", ans);
}

```

* Bisection

$$\begin{array}{c} \overline{a, b} \\ f \end{array} \rightarrow c = \frac{(a+b)}{2} \rightarrow \begin{array}{c} \overline{f(c)} \times \overline{f(a)} \\ f(c) \times f(b) \end{array} < 0 \rightarrow b = c$$

$$\begin{array}{c} \overline{f(c)} \times \overline{f(a)} \\ f(c) \times f(b) \end{array} < 0 \rightarrow a = c$$

```

//Bisection Method

#include<stdio.h>
#include<math.h>

```

```

#define f(x) (exp(x)-3*x)

int main(){
    double a,b;
    scanf("%lf", &a);
    scanf("%lf", &b);

    while(true){
        double c = (a+b)/2;
        if(f(a)*f(c) < 0) b=c;
        else a=c;
        if(fabs((a)-(b)) <= 0.0001) break;
    }
    printf("%lf", a);
}

#include<stdio.h>
#include<math.h>

#define f(x) (x*log10(x)-1.2)

int main(){
    double a,b;
    scanf("%lf", &a);
    scanf("%lf", &b);

    while(true){
        double c = (a+b)/2;
        if(f(a)*f(c) < 0) b=c;
        else a=c;
        if(fabs((a)-(b)) <= 0.0001) break;
    }
    printf("%lf", a);
}

#include<stdio.h>
#include<math.h>

#define f(x) (x*x*x-6*x+4)

int main(){
    double a,b;
    scanf("%lf", &a);
    scanf("%lf", &b);

    while(true){
        double c = (a+b)/2;
        if(f(a)*f(c) < 0) b=c;
        else a=c;
        if(fabs((a)-(b)) <= 0.0001) break;
    }
    printf("%lf", a);
}

```

* $\overline{a, b}$
 f $\rightarrow c = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$

$f(c) \cdot f(a) < 0 \Rightarrow b = c$
 Otherwise $\rightarrow a = c$

```
// False Position

#include<stdio.h>
#include<math.h>

#define f(x) (x*x*x-6*x+4)

int main(){
    double a,b;
    scanf("%lf", &a);
    scanf("%lf", &b);
    double ans=a;
    while(true){
        double c = ((a*f(b) - b*f(a))/(f(b)-f(a)));
        if(f(a)*f(c) < 0) b=c;
        else a=c;
        ans=c;
        if(fabs(f(a)-f(b)) <= 0.0001) break;
    }
    printf("%lf", ans);
}
```

*
 x_0
 f
 f'

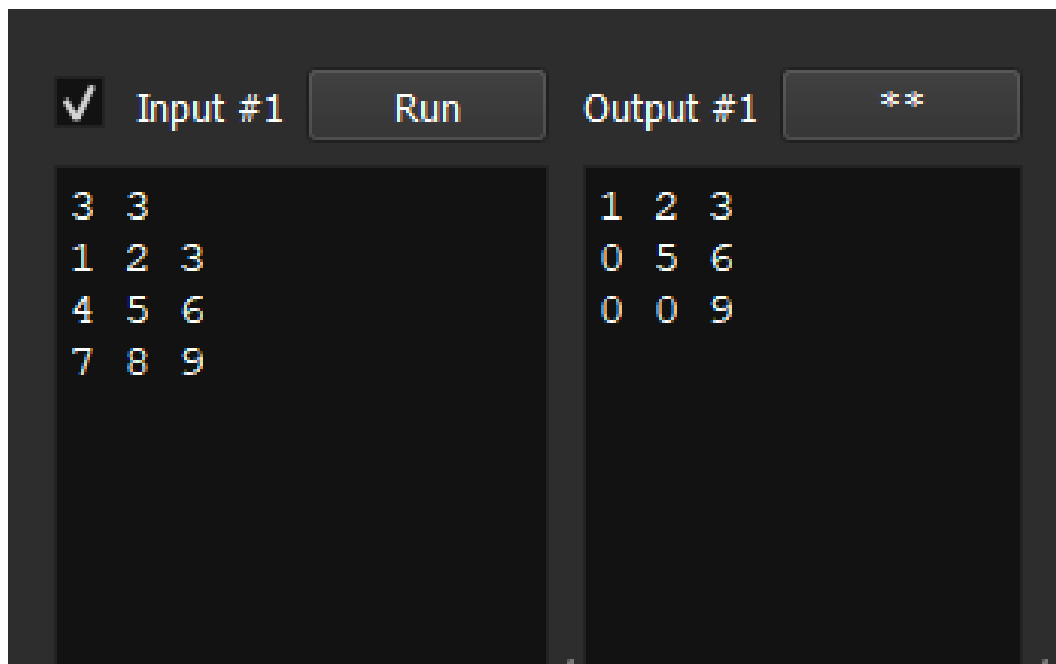
$\rightarrow x = x_0 - \frac{f(x_0)}{f'(x)}$

$[|x - x_0| \leq 10^{-4}]$

```
// Newton Raphson

#include<stdio.h>
#include<math.h>

#define f(x) (x*x*x-6*x+4)
#define diff(x) (3*x*x - 6)
int main(){
    double x0;
    scanf("%lf", &x0);
    double x=x0;
    while(true){
        x = x0 - (f(x)/diff(x));
        if(fabs(f(x)-f(x0)) <= 0.0001) break;
        x0=x;
    }
    printf("%lf", x);
}
```



```
// Upper-triangular method

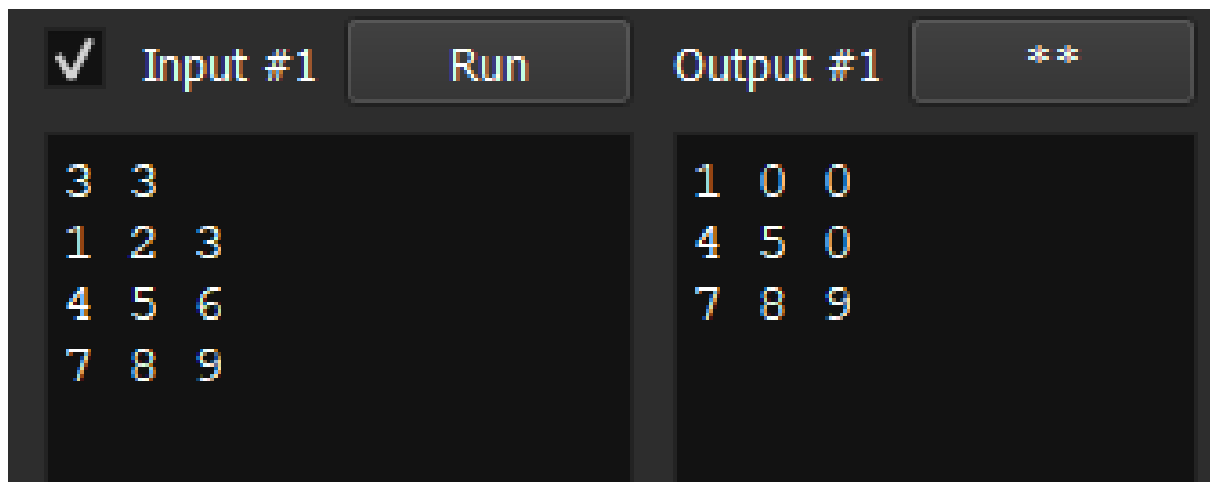
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    int a[n][m];

    for(ll i=0; i<n; i++)
        for(ll j=0; j<m; j++)
            scanf("%d", &a[i][j]);

    for(ll i=0; i<n; i++){
        for(ll j=0; j<i; j++) printf("0 ");
        for(ll j=i; j<m; j++){
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
}
```



```
// lower triangular
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    int a[n][m];

    for(ll i=0; i<n; i++)
        for(ll j=0; j<m; j++)
            scanf("%d", &a[i][j]);

    for(ll i=0; i<n; i++){
        for(ll j=0; j<=i; j++){
            printf("%d ", a[i][j]);
        }
        for(ll j=i+1; j<m; j++) printf("0 ");
        printf("\n");
    }
}
```




```
//Diagonal Matrix
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    int a[n][m];

    for(ll i=0; i<n; i++)
        for(ll j=0; j<m; j++)
            scanf("%d", &a[i][j]);

    for(ll i=0; i<n; i++){
        for(ll j=0; j<m; j++){
            if(i==j) printf("%d ", a[i][j]);
            else printf("0 ");
        }
        printf("\n");
    }
}
```

$$a[0][i] \{ a[1][(i+1)\%n] \times a[2][(i+1)\%n] - a[1][(i+2)\%n] \times a[2][(i+1)\%n] \}$$

```
// Deteminant

#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    int a[n][m];

    for(ll i=0; i<n; i++)
        for(ll j=0; j<m; j++)
            scanf("%d", &a[i][j]);

    double determinant=0;
    for(ll i=0; i<n; i++)
        determinant += a[0][i]*(a[1][(i+1)%n]*a[2][(i+2)%n] - a[1][(i+2)%n]*a[2][(i+1)%n]);

    printf("%lf\n", determinant);
}
```

Input #1	Run	Output #1
<pre>3 3 3 1 2 2 -3 -1 1 2 1</pre>		<pre>3 2 1 1 -3 2 2 -1 1</pre>

```

//Transpose Matrix
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
    int a[n][m];

    for(ll i=0; i<n; i++)
        for(ll j=0; j<m; j++)
            scanf("%d", &a[i][j]);

    for(ll i=0; i<n; i++){
        for(ll j=0; j<n; j++){
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }
}

```

$$1) \text{adj}[i][j] = a[(i+1) \cdot n][(j+1) \cdot n] \times a[(i+2) \cdot n][(j+2) \cdot n] - a[(i+1) \cdot n][(j+2) \cdot n] \times a[(i+2) \cdot n][(j+1) \cdot n];$$

$$2) \text{transpose}(\text{adj}) = \text{adjoint}$$

```

//Adjoint Matrix
#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);

```

```

int a[n][m];

for(ll i=0; i<n; i++)
    for(ll j=0; j<m; j++)
        scanf("%d", &a[i][j]);

int adj[n][m];
for(ll i=0; i<n; i++){
    for(ll j=0; j<n; j++){
        adj[i][j] = a[(i+1)%n][(j+1)%n]*a[(i+2)%n][(j+2)%n]
                    - a[(i+1)%n][(j+2)%n]*a[(i+2)%n][(j+1)%n];
    }
}

for(ll i=0; i<n; i++)
{
    for(ll j=0; j<m; j++){
        printf("%d ", adj[j][i]);
    }
    printf("\n");
}
}

```

1)
$$\frac{adj[i][j]}{\text{determinant}}$$

```

//Inverse Matrix

#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n,m;
    scanf("%d %d", &n, &m);
}

```

```

int a[n][m];

for(ll i=0; i<n; i++)
    for(ll j=0; j<m; j++)
        scanf("%d", &a[i][j]);

double determinant = 0;
for(ll i=0; i<n; i++)
    determinant += a[0][i]*(a[1][(i+1)%n]*a[2][(i+2)%n] - a[1][(i+2)%n]*a[2][(i+1)%n]);

double adj[n][m];
for(ll i=0; i<n; i++)
    for(ll j=0; j<m; j++)
        adj[i][j] = a[(i+1)%n][(j+1)%n]*a[(i+2)%n][(j+2)%n]
                    - a[(i+1)%n][(j+2)%n]*a[(i+2)%n][(j+1)%n];

for(ll i=0; i<n; i++){
    for(ll j=0; j<m; j++){
        printf("%0.4lf ", adj[j][i]/determinant);
    }
    printf("\n");
}
}

```

```

// Gauss Elimination

#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n;
    scanf("%d", &n);
    double a[n+1][n+2];

    for(ll i=1; i<=n; i++){
        for(ll j=1; j<=n; j++){
            scanf("%lf", &a[i][j]);
        }
    }

    for(ll i=1; i<=n; i++)
        scanf("%lf", &a[i][n+1]);

    double c;
    for(ll j=1; j<=n; j++){
        for(ll i=1; i<=n; i++){

```

```

        if(i > j){
            c=a[i][j]/a[j][j];
            for(ll k=1; k<=n+1; k++){
                a[i][k]=a[i][k]-c*a[j][k];
            }
        }
    }
}

double x[n+1];
x[n]=a[n][n+1]/a[n][n];
for(ll i=n-1; i>=1; i--){
    double sum=0;
    for(ll j=i+1; j<=n; j++){
        sum += a[i][j]*x[j];
    }
    x[i]=(a[i][n+1]-sum)/a[i][i];
}

for(ll i=1; i<=n; i++) printf("%lf\n", x[i]);
}

```

```

// Matrix Inversion Method

#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    int n;
    scanf("%d", &n);
    double a[n][n], b[n];

    for(ll i=0; i<n; i++){
        for(ll j=0; j<n; j++){
            scanf("%lf", &a[i][j]);
        }
    }

    for(ll i=0; i<n; i++)
        scanf("%lf", &b[i]);

    double determinant=0;
    for(ll i=0; i<n; i++)
        determinant += a[0][i]*(a[1][(i+1)%n]*a[2][(i+2)%n]
            - a[1][(i+2)%n]*a[2][(i+1)%n]);

    double adj[n][n];
    for(ll i=0; i<n; i++)

```

```

    for(ll j=0; j<n; j++)
        adj[i][j] = a[(i+1)%n][(j+1)%n]*a[(i+2)%n][(j+2)%n]
            - a[(i+1)%n][(j+2)%n]*a[(i+2)%n][(j+1)%n];

    double adjoint[n][n];
    for(ll i=0; i<n; i++)
        for(ll j=0; j<n; j++)
            adjoint[j][i]=(adj[i][j]);

    for(ll i=0; i<n; i++){
        double sum=0;
        for(ll j=0; j<n; j++){
            sum += b[j]*adjoint[i][j];
        }
        printf("%.4lf\n", (sum/determinant));
    }
}

```

Trapezoidal Rule

$$\int_a^b f(x)dx = h\left(\frac{y_0+y_n}{2} + y_1 + \dots + y_{n-1}\right)$$

```

// Trapezoidal Rule

#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    ll n; scanf("%d", &n);
    double a[n], b[n];

    for(ll i=0; i<n; i++)
        scanf("%lf %lf", &a[i], &b[i]);
    double xi, xf, h;
    scanf("%lf %lf %lf", &xi, &xf, &h);

    double ans=(b[0]+b[n-1])/2;
    for(ll i=1; i<n-1; i++) ans += b[i];
    ans *= h;

    printf("%lf", ans);
}

```

Simpson one-third rule

$$\int_a^b f(x)dx = \frac{h}{3}(y_0 + y_n + 4(y_1 + y_3 + \dots) + 2(y_2 + y_4 + \dots))$$

Applicable for only **even intervals**.

```
// Simpson one-third rule

#include<stdio.h>
#include<math.h>

#define ll int

int main(){
    ll n; scanf("%d", &n);
    double a[n], b[n];

    for(ll i=0; i<n; i++)
        scanf("%lf %lf", &a[i], &b[i]);
    double xi, xf, h;
    scanf("%lf %lf %lf", &xi, &xf, &h);

    double ans=(b[0]+b[n-1]);
    for(ll i=1; i<n-1; i+=2) ans += 4*b[i];
    for(ll i=2; i<n-1; i+=2) ans += 2*b[i];
    ans *= (h/3);

    printf("%lf", ans);
}
```

```
//Newton Forward Interpolation

#include <stdio.h>
#include <math.h>

int main()
{
    int n;
    printf("The number of x and f(x)");
    scanf("%d", &n);
    double x[n], y[n], xi;
```



```

printf("Write the number of x:");
for (int i = 0; i < n; i++)
{
    scanf("%lf", &x[i]);
}
printf("Write the number of f(x):");
for (int i = 0; i < n; i++)
{
    scanf("%lf", &y[i]);
}

printf("X = ");
scanf("%lf", &xi);

double da[n][n];
for (int j = 0; j < n; j++)
{
    da[j][0] = y[j];
}

for (int i = 1; i < n; i++)
{
    for (int j = 0; j < n - i; j++)
    {
        da[j][i] = (da[j + 1][i - 1] - da[j][i - 1]);
    }
}

double p = (xi - x[0]) / (x[1] - x[0]);
double ans = y[0];
double fact=1;
for (int i = 1; i < n; i++)
{
    double localP = p * 1.00;
    fact *= 1.00;

    for (int j = 1; j < i; j++)
    {
        localP *= (p - j);
    }
    ans += (localP / fact) * da[0][i];
}

printf(" %lf ", ans);

return 0;
}

```

```

//Newton Backward

#include <stdio.h>

```

```

#include <math.h>

int main()
{
    int n;
    printf("The number of x and f(x)");
    scanf("%d", &n);
    double x[n], y[n], xi;
    printf("Write the number of x:");
    for (int i = 0; i < n; i++)
    {
        scanf("%lf", &x[i]);
    }
    printf("Write the number of f(x):");
    for (int i = 0; i < n; i++)
    {
        scanf("%lf", &y[i]);
    }

    printf("X = ");
    scanf("%lf", &xi);

    double da[n][n];
    for (int j = 0; j < n; j++)
    {
        da[j][0] = y[j];
    }

    for (int i = 1; i < n; i++)
    {
        for (int j = 0; j < n - i; j++)
        {
            // printf(" %lf %lf ", da[j + 1][i - 1], da[j][i - 1]);
            da[j][i] = (da[j + 1][i - 1] - da[j][i - 1]);
            // printf(" %lf \n ", da[j][i]);
        }
    }

    double p = (xi - x[n - 1]) / (x[1] - x[0]);
    double ans = y[n - 1];
    for (int i = n - 1; i >= 1; i--)
    {
        double localP = p * 1.00;
        double fact = 1.00;

        for (int j = 1; j <= n - i - 1; j++)
        {
            fact = fact * j;
        }
        for (int j = 1; j < n - i - 1; j++)
        {
            localP *= (p - j);
        }
        ans += (localP / fact) * da[n - i - 1][i];
    }
}

```

```

printf("%lf ", ans);

return 0;
}

```

```

//Newton Devided Difference

#include <stdio.h>
#include<stdlib.h>
#include <math.h>

int main()
{
    int n;
    //taking the total number of x
    printf("Enter number of x and y or f(x)\n");
    scanf("%d", &n);
    double x[n], y[n], X;

    // taking the value of X
    printf("jot down the value of x\n");
    for (int i = 0; i < n; i++)
        scanf("%lf", &x[i]);

    // taking the value of y/f(x)
    printf("write down the number of y or f(x):\n");
    for (int i = 0; i < n; i++)
        scanf("%lf", &y[i]);

    //taking the value of question
    printf("The X you want to get = ");
    scanf("%lf", &X);

    double table[n][n];
    for (int j = 0; j < n; j++)
        table[j][0] = y[j];

    for (int i = 1; i < n; i++)
        for (int k = 0; k < n - i; k++)
            table[k][i] = (table[k + 1][i - 1] - table[k][i - 1]) / (x[i + k] - x[k]);

    double ans = y[0];
    for (int i = 1; i < n-1; i++)
    {
        double ansa = 1.00;
        for (int k = 0; k <= i-1; k++)
            ansa *= (X - x[k]);
        ans += (ansa)*table[0][i];
    }
}

```

```

//printing ans
printf("Ans: %lf ", ans);
return 0;
}

```

```

//Numerical Differentiation

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int n;
int main()
{
    //taking the total number of x
    printf("Enter number of x and y or f(x)\n");
    scanf("%d", &n);
    double x[n], y[n], X;

    // taking the value of X
    printf("jot down the value of x\n");
    for (int i = 0; i < n; i++)
        scanf("%lf", &x[i]);

    // taking the value of y/f(x)
    printf("write down the number of y or f(x):\n");
    for (int i = 0; i < n; i++)
        scanf("%lf", &y[i]);

    //taking the value of question
    printf("The X you want to get = ");
    scanf("%lf", &X);

    int Xindex;

    for(int i=0; i<n; i++){
        if(X == x[i]) Xindex=i;
    }

    double table[n][n];
    for (int j = 0; j < n; j++)
    {
        table[j][0] = y[j];
    }

    for (int i = 1; i < n; i++)
        for (int j = 0; j < n - i; j++)
            table[j][i] = (table[j + 1][i - 1] - table[j][i - 1]);

    double ans = 0.000000;
}

```

```

for (int i = 0; i < n; i++)
{
    if(i%2 == 1)
        ans-=(1/(i+1))*(table[Xindex-i][i]);

    else
        ans+=(1/(i+1))*(table[Xindex-i][i]);
}

printf("Ans for the 3rd derivative : %lf ", ans);

return 0;
}

```

```

//Lagrange Interpolation

#include <stdio.h>
#include <math.h>

int main()
{
    int n;
    printf("The number of x and f(x)");
    scanf("%d", &n);
    double x[n], y[n], xi;
    printf("Write the number of x:");
    for (int i = 0; i < n; i++)
    {
        scanf("%lf", &x[i]);
    }
    printf("Write the number of f(x):");
    for (int i = 0; i < n; i++)
    {
        scanf("%lf", &y[i]);
    }

    printf("X = ");
    scanf("%lf", &xi);

    double da[n][n];
    for (int j = 0; j < n; j++)
    {
        da[j][0] = y[j];
    }

    double ans = 0;
    for (int i = 0; i < n; i++)
    {
        double localAns = 1;
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                continue;

```

```

        localAns *= (xi - x[j]);
        localAns /= (x[i] - x[j]);
    }
    ans += localAns * y[i];
}
printf(" %lf ", ans);

return 0;
}

```

```

//Iterative Method

#include <stdio.h>
#include <math.h>

#define f(x) (x * x * x + x * x - 1)
#define phi(x) 1 / sqrt(1 + x)
#define diffPhi(x) (0.5 / sqrt(1 + x))

/*
y = x^3 - 2x - 5, is unsolvable by Iteration method.
idky.

#define f(x) (x * x * x - 2* x - 5)
#define phi(x) 5/(x*x - 2)
#define diffPhi(x)(10*x/((x*x - 2)*(x*x - 2)))

*/
int main()
{
    double a = 0, b = 0;
    double x1 = 0, x2 = 0;

    // Step 1 : finding the value of x1 and x2
    while (1)
    {
        /*
        ekhane dui bhabhe check kortechi,
        dhoren,
        1) (0 1), (1,2) , ...
        2) (0, -1), (-1, -2), ....
        ig, yk why.
        */
        if (f(a) * f((a - (double)1.0000)) < 0)
        {
            printf("%lf %lf", f(a), f((a - (double)1.0000)));
            x1 = a;
            x2 = a - 1;

            break;
        }
        if (f(b) * f((b + (double)1.0000)) < 0)
        {

```

```

        x1 = b;
        x2 = b + 1;
        break;
    }
    a--;
    b++;
}
// Step 2 & 3 : finding the root
// we'll just jot down the steps which are written on the blog
double ans = 1e9; // Just assume a number which can never be the answer
double x0 = (x1 + x2) / 2;

if (abs(diffPhi(x0)) < 1)
{
    while (1)
    {
        double xn = phi(x0);
        if (fabs(xn - x0) <= 0.001)
        {
            ans = xn;
            break;
        }
        x0 = xn;
    }
}
else
{
    printf("NOT FOUND");
    return 0;
}

// Printing the answer
printf("%lf", ans);
return 0;
}

```

```

//Secant Method

#include <stdio.h>
#include <math.h>
#define f(x) (x * x * x - 2 * x - 5)

int main()
{
    double a = 0, b = 0;
    double x1 = 0, x2 = 0;

    // Step 1 : finding the value of x1 and x2
    while (1)
    {
        /*
            ekhane dui bhabe check kortechi,
            dhoren,

```

```

1) (0 1), (1,2) , ...
2) (0, -1), (-1, -2), ....
ig, yk why.
*/
if (f(a) * f((a - (double)1.0000)) < 0)
{
    printf("%lf %lf", f(a), f((a - (double)1.0000)));
    x1 = a;
    x2 = a - 1;

    break;
}
if (f(b) * f((b + (double)1.0000)) < 0)
{
    x1 = b;
    x2 = b + 1;
    break;
}
a--;
b++;
}
// Step 2 & 3 : finding the root

// we'll just jot down the steps which are written on the blog
double ans = 1e9; // Just assume a number which can never be the answer
while (1)
{
    double prevAns = ans;
    double x = ((x1 * f(x2) - x2 * f(x1)) / (f(x2) - f(x1)));
    ans = x;
    x1 = x2;
    x2 = x;
    // Calculating the the  iteration of last two answer
    if (fabs((prevAns) - (ans)) <= 0.001)
    {
        break;
    }
}

// Printing the answer
printf("%lf", ans);
return 0;
}

```