Bisection Method

```
#include <bits/stdc++.h>
using namespace std;
#define error 0.09

double func(double x)
{
   return x * x * x + 2 * x * x - x + 8;
}
void bisection(double a, double b)
{
   if (func(a) * func(b) >= 0)
   {
      cout << "You have not assumed right a and b"<<endl;
   }
   double c = a;
   int i = 0;
   while ((b - a) >= error)
   {
      c = (a + b) / 2;
      if (func(c) == 0.0)
         break;
      else if (func(c) * func(a) < 0)
         b = c;
      else
         a = c;
      i++;
   }

   cout << "The value of root is : " << c << endl;
   cout << "Number of iterations: " << i << endl;
}

int main()
{
   // Initial values assumed
   double a =-4, b =4;
   bisection(a, b);
   return 0;
}
```

False Position

```
#include <bits/stdc++.h>
using namespace std;
#define error 0.09

double func(double x)
{
   return x * x * x + 2 * x * x - x + 8;
}

void falsePosition(double a, double b)
{
   if (func(a) * func(b) >= 0)
   {
      cout << "You have not assumed right a and b" << endl;
      return;
   }

   double c = a;
   int i = 0;
   double fa, fb, fc;
   while ((b - a) >= error)
   {
      fa = func(a);
      fb = func(b);
      c = (a * fb - b * fa) / (fb - fa);

      fc = func(c);

      if (fc == 0.0 || abs(fc) < error)
         break;

      if (fc * fa < 0)
         b = c;
      else
         a = c;

      i++;
   }

   cout << "The value of root is : " << c << endl;
   cout << "Number of iterations: " << i << endl;
}

int main()
{
   double a = -4, b = 4;
   falsePosition(a, b);
   return 0;
}
```

Secant Method

```
#include <bits/stdc++.h>
using namespace std;
#define error 0.09

double func(double x)
{
   return exp(-x) -x;
}

double secant(double x0, double x1)
{
   int i = 0;
   double x2, f0, f1, f2;

   while (i < 10)
   {
      f0 = func(x0);
      f1 = func(x1);

      if (abs(f1 - f0) < error) return x1;

      x2 = x1 - (f1 * (x1 - x0)) / (f1 - f0);

      x0 = x1;
      x1 = x2;
      i++;
   }

   cout << "Method did not converge!" << endl;
   return x1;
}

int main()
{
   // Initial values assumed
   double x0 = 0, x1 = 1;
   double root = secant(x0, x1);

   cout << "The value of root is: " << root << endl;

   return 0;
}
```

Trapizoidal

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

double f(double x)
{
   return x*x-x+2;
}

// a=0 b=2 n=4 ans= 4.75
// multiple trapizoidal
void solve()
{
   double a,b,n,s;
   cin>>a>>b>>n;

   double h = (b-a)/n;

   s = f(a) + f(b);

   for(int i=1 ; i<n ; i++)
   {
      s+= 2*f(a+i*h);
   }
   double ans = (h*s)/2;
   cout<<ans<<endl;
}
```

```
//single trapizoidal
void solve2()
{
   double a, b;
   cin >> a >> b;
   cout << (b-a)*((f(a)+f(b))/2);
}

int main()
{
   solve();
   solve2();
}
```

Simphson

```
#include<bits/stdc++.h>

using namespace std;

// 0.2 + 25x – 200x^2 + 675x^3 – 900x^4 + 400x^5

double fun(double x)

{

   return 0.2+25*x-200*x*x+675*x*x*x-900*pow(x,4)+400*pow(x,5);

}

void fun_3_8()

{

   double a = 0, b = 2; // 4.75

   cin >> a >> b;

   double h = (b-a)/3;

   vector <double> v(100);

   int j = 0;

   for (double i = a; i <= b; i += h)

   {

      v[j++] = fun(i);

   }

   double I = ((3*h)/8) *(v[0] + 3*v[1] + 3*v[2] + v[3]);

   cout << I << endl;

}

void fun_1_3()

{

   double a = 0, b = 2; // 4.75

   cin >> a >> b;

   double h = (b-a)/2;

   vector <double> v(100);

   int j = 0;

   for (double i = a; i <= b; i += h)

   {

      v[j++] = fun(i);

   }

   double ans = 0;

   ans += (2*h) * (v[0]+4*v[1]+v[2]);

   cout << ans/6 << endl;

}

void fun_1_3_mul()

{

   double a = 0, b = 2, n = 4; // 4.75

   cin >> a >> b >> n;

   double h = (b-a)/n;

   vector <double> v(100);

   int j = 0;

   for (double i = a; i <= b; i += h)

   {

      v[j++] = fun(i);

   }

   j = 0;

   double ans = 0;

   for (int i = 0; i < n; i++)
```

```
    {
        //ans += h*((v[j]+v[j+1])/2);
        ans += ((v[j]+v[j+1]*4+v[j+2]));
        j += 2;
    }
    cout << ans*((2*h)/6) << endl;
}
int main()
{
    fun_3_8();
}
```