Experiment No.: 01
Name of the Experiment: Implementation of Bisection Method

Theory:

The method is applicable for numerically solving the equation $f(x) = 0$ for the real variable x, where f is a continuous function defined on an interval $[a, b]$ and where $f(a)$ and $f(b)$ have opposite signs. In this case a and b are said to bracket a root since, by the intermediate value theorem, the continuous function f must have at least one root in the interval (a, b).

At each step the method divides the interval in two by computing the midpoint

$$x = \frac{a+b}{2}$$

of the interval and the value of the function $f(x)$ at that point. Unless x is itself a root (which is very unlikely, but possible) there are now only two possibilities: either $f(a)$ and $f(x)$ have opposite signs and bracket a root, or $f(x)$ and $f(b)$ have opposite signs and bracket a root. The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

Explicitly, if $f(a)$ and $f(x)$ have opposite signs, then the method sets x as the new value for b, and if $f(b)$ and $f(x)$ have opposite signs then the method sets x as the new a. (If $f(x)=0$ then x may be taken as the solution and the process stops.) In both cases, the new $f(a)$ and $f(b)$ have opposite signs, so the method is applicable to this smaller interval.

Each iteration performs these steps:

1. Calculate x, the midpoint of the interval, $x = \frac{a+b}{2}$.
2. Calculate the function value at the midpoint, $f(x)$.
3. If convergence is satisfactory (that is, $x - a$ is sufficiently small, or $|f(x)|$ is sufficiently small), return x and stop iterating.
4. Examine the sign of $f(x)$ and replace either $(a, f(a))$ or $(b, f(b))$ with $(x, f(x))$ so that there is a zero crossing within the new interval.

Code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;

double f(double x)
{
    return ((x*x*x) - (2*x) - 5);
}

int main(void)
{
    double a = 2,b = 3;
    double x0;
    double x=0;
```

```
    int n=1;

    printf(" n|      a       |      b       |      x       |     f(x)     \n");
    printf("--------------------------------------------------------\n");
    while(1)
    {
        x0=x;
        x=(a+b)/2;
        if(abs(x0-x)>=0.0001)
        {
            if(f(a)*f(x)>0)
            {
                printf("%2d|%3.10f|%3.10f|%3.10f|%3.10f\n",n,a,b,x,f(x));
                a=x;
            }
            else
            {
                printf("%2d|%3.10f|%3.10f|%3.10f|%3.10f\n",n,a,b,x,f(x));
                b=x;
            }
            n++;
            printf("------------------------------------------------------
-\n");
        }
        else
            break;
    }
    printf("Answer: ");
    cout<<x<<endl;
}
```

Output:

```
 n|      a       |      b       |      x       |     f(x)
-------------------------------------------------------
 1|2.0000000000|3.0000000000|2.5000000000|5.6250000000
-------------------------------------------------------
 2|2.0000000000|2.5000000000|2.2500000000|1.8906250000
-------------------------------------------------------
 3|2.0000000000|2.2500000000|2.1250000000|0.3457031250
-------------------------------------------------------
 4|2.0000000000|2.1250000000|2.0625000000|-0.3513183594
-------------------------------------------------------
 5|2.0625000000|2.1250000000|2.0937500000|-0.0089416504
-------------------------------------------------------
 6|2.0937500000|2.1250000000|2.1093750000|0.1668357849
-------------------------------------------------------
 7|2.0937500000|2.1093750000|2.1015625000|0.0785622597
-------------------------------------------------------
 8|2.0937500000|2.1015625000|2.0976562500|0.0347142816
-------------------------------------------------------
 9|2.0937500000|2.0976562500|2.0957031250|0.0128623322
-------------------------------------------------------
10|2.0937500000|2.0957031250|2.0947265625|0.0019543478
-------------------------------------------------------
11|2.0937500000|2.0947265625|2.0942382812|-0.0034951492
-------------------------------------------------------
12|2.0942382812|2.0947265625|2.0944824219|-0.0007707752
-------------------------------------------------------
13|2.0944824219|2.0947265625|2.0946044922|0.0005916927
-------------------------------------------------------
Answer: 2.09454

Process returned 0 (0x0)   execution time : 0.392 s
Press any key to continue.
```

Discussion:

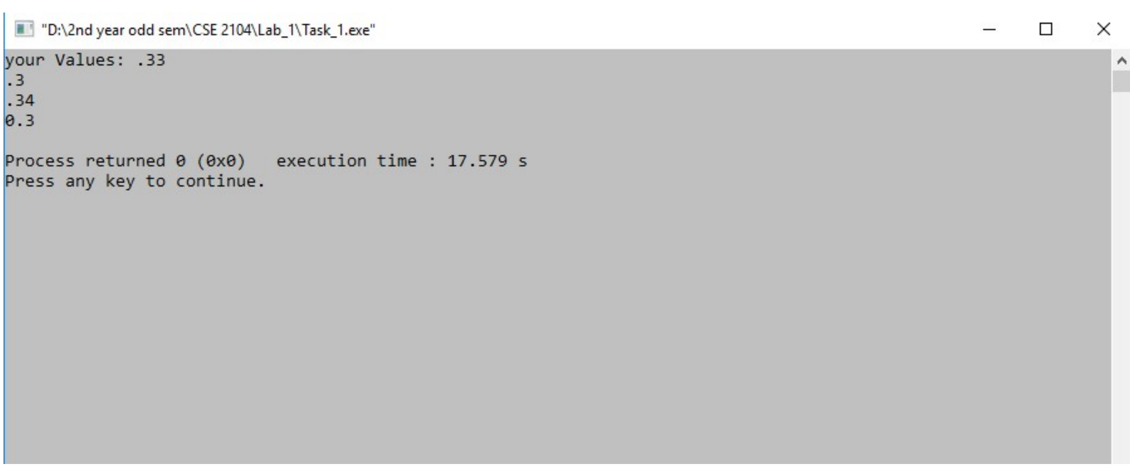
When implementing the method on a computer, there can be problems with finite precision, so there are often additional convergence tests or limits to the number of iterations. Although f is continuous, finite precision may preclude a function value ever being zero. Additionally, the difference between a and b is limited by the floating point precision; i.e., as the difference between a and b decreases, at some point the midpoint of $[a, b]$ will be numerically identical to (within floating point precision of) either a or b.

Task: 01

Code:

```
#include<iostream>
#include<cstdlib>
#include<cmath>
using namespace std;

int main(void)
{
    double x1,x2,x3;
    double xt;
    double xa[3];
    double nearest;

    xt=1/3;
    cout<<"your Values: ";
    cin>>x1>>x2>>x3;

    xa[0]=abs(xt-x1);
    xa[1]=abs(xt-x2);
    xa[2]=abs(xt-x3);

    if(xa[0]<xa[1]&&xa[0]<xa[2])
        cout<<x1<<endl;
    else if(xa[1]<xa[0]&&xa[1]<xa[2])
        cout<<x2<<endl;
    else
        cout<<x3<<endl;
}
```

Output:



Task: 2

Code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;

int main(void)
{
    double xt;
    double x;
    double xa,xr,xp;

    xt=sqrt(2);

    cin>>x;

    xa=abs(xt-x);
    xr=xa/xt;
    xp=xr*100;

    cout<<”Xa: “<<xa<<endl;
    cout<<”Xr: “<<xr<<endl;
    cout<<”Xp: “<<xp<<endl;
}
```

Output: