Numerical Methods HW #02

1. 선형 보간법을 사용하여 다음 방정식의 가장 작은 양의 근들을 유효숫자 5자리 이내에 있도록 결정하라. (선형 보간법으로 모든 근을 구하고 가장 작은 양의 근을 선택했다)

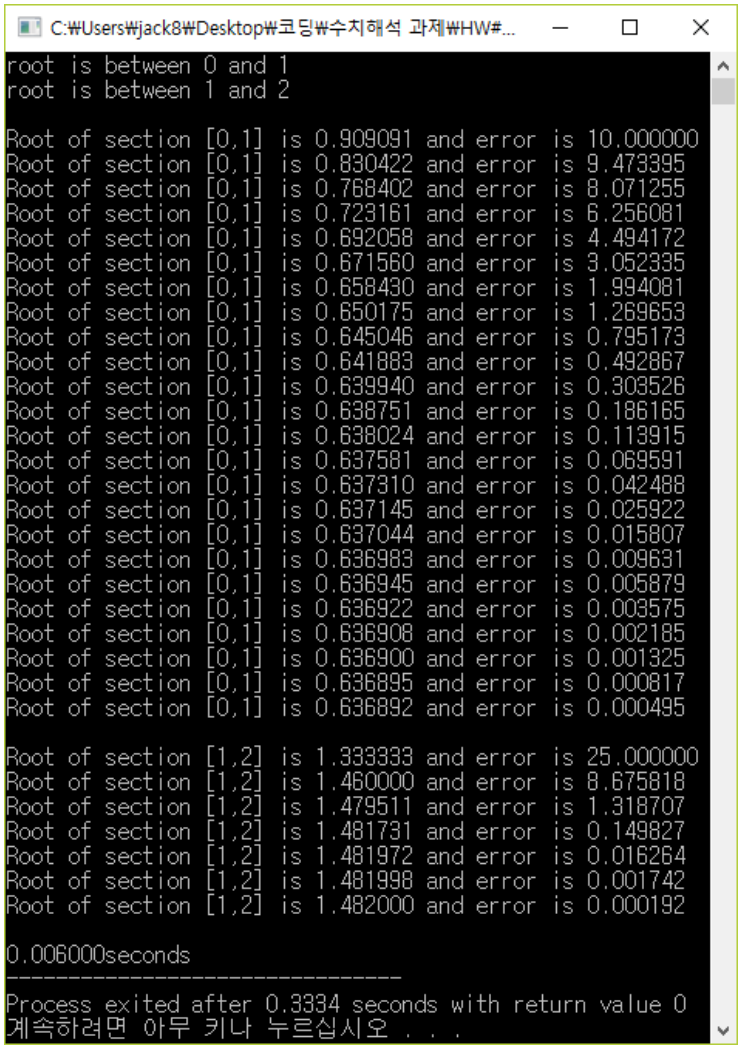
참 값을 Wolfram Alpha로 구해서 구한 값과 비교했다.

(a) $x^4 - 8x^3 + 24x^2 - 28x + 10 = 0$ 참값=0.63689, 1.4820

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return pow(x,4)-8*pow(x,3)+24*pow(x,2)-28*x+10;
}
int main()
{
	//Incremental-Search Method(증가 탐색법)
	int i,j;
	j=0;
	double a,b,c;
	double x[4];
	for(i=-20;i<=20;i++)
	{
		if(func(i)*func(i+1)<0)
		{
			x[j]=(double)i;
			printf("root is between %d and %d\n",i,i+1);
			j++;
		}
	}
	printf("\n");
	//Linear Interpolation Method(선형 보간법)
	double error=0.5/pow(10,3);
	clock_t start,end;
	start=clock();
	for(i=0;i<=1;i++)
	{
		a=x[i]; b=x[i]+1;
		for(j=0;j<=100;j++)
		{
			c=b-(b-a)/(func(b)-func(a))*func(b);
			if(func(a)*func(c)<0)
			{
				printf("Root of section [%.0lf,%.0lf] is %lf and error
is %lf\n",x[i],x[i]+1,c,fabs((b-c)/c*100));
				if(j!=0)
				{if(fabs((b-c)/c*100)<error) break;}
				b=c;
			}
			else
			{
				printf("Root of section [%.0lf,%.0lf] is %lf and error
is %lf\n",x[i],x[i]+1,c,fabs((a-c)/c*100));
				if(j!=0)
				{if(fabs((a-c)/c*100)<error) break;}
				a=c;
			}
		}
		printf("\n");
	}
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



(b) $xe^x - 4x - 5 = 0$ 참값=-1.33777, 1.89326

<Code>

```c
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return x*exp(x)-4*x-5;
}
int main()
{
	//Incremental-Search Method(증가 탐색법)
	int i,j;
	j=0;
	double a,b,c;
	double x[4];
	for(i=-40;i<=40;i++)
	{
		if(func(i)*func(i+1)<0)
		{
			x[j]=(double)i;
			printf("root is between %d and %d\n",i,i+1);
			j++;
		}
	}
	printf("\n");
```

```
        //Linear Interpolation Method(선형 보간법)
        double error=0.5/pow(10,3);
        clock_t start,end;
        start=clock();
        for(i=0;i<=1;i++)
        {
                a=x[i]; b=x[i]+1;
                for(j=0;j<=100;j++)
                {
                        c=b-(b-a)/(func(b)-func(a))*func(b);
                        if(func(a)*func(c)<0)
                        {
                                printf("Root of section [%.0lf,%.0lf] is %lf and error
is %lf\n",x[i],x[i]+1,c,fabs((b-c)/c*100));
                                if(j!=0)
                                {if(fabs((b-c)/c*100)<error) break;}
                                b=c;
                        }
                        else
                        {
                                printf("Root of section [%.0lf,%.0lf] is %lf and error
is %lf\n",x[i],x[i]+1,c,fabs((a-c)/c*100));
                                if(j!=0)
                                {if(fabs((a-c)/c*100)<error) break;}
                                a=c;
                        }
                }
                printf("\n");
        }
        end=clock();
        printf("%lfseconds",double(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

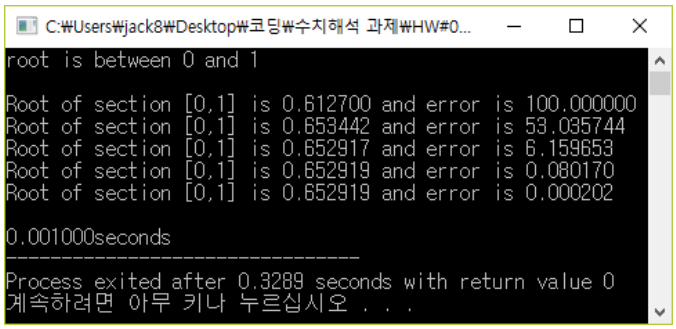
```
C:₩Users₩jack8₩Desktop₩코딩₩수치해석 과제₩HW#02₩...
root is between -2 and -1
root is between 1 and 2

Root of section [-2,-1] is -1.333856 and error is 25.029410
Root of section [-2,-1] is -1.337737 and error is 0.290064
Root of section [-2,-1] is -1.337768 and error is 0.002310
Root of section [-2,-1] is -1.337768 and error is 0.000020

Root of section [1,2] is 1.779386 and error is 43.800836
Root of section [1,2] is 1.882925 and error is 5.498820
Root of section [1,2] is 1.892356 and error is 0.498423
Root of section [1,2] is 1.893178 and error is 0.043371
Root of section [1,2] is 1.893249 and error is 0.003759
Root of section [1,2] is 1.893255 and error is 0.000326

0.003000seconds
--------------------------------
Process exited after 0.3507 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

(c) $x - e^{-x^2} = 0$ 참값=0.65292

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return x-exp(-pow(x,2));
}
int main()
{
	//Incremental-Search Method
	int i,j;
	j=0;
	double a,b,c;
	double x[4];
	for(i=-40;i<=40;i++)
	{
		if(func(i)*func(i+1)<0)
		{
			x[j]=(double)i;
			printf("root is between %d and %d\n",i,i+1);
			j++;
		}
	}
	printf("\n");
	//Linear Interpolation Method
	double error=0.5/pow(10,3);
	clock_t start,end;
	start=clock();
	a=x[i]; b=x[i]+1;
	for(j=0;j<=100;j++)
	{
		c=b-(b-a)/(func(b)-func(a))*func(b);
		if(func(a)*func(c)<0)
		{
			printf("Root of section [%.0lf,%.0lf] is %lf and error
is %lf\n",x[i],x[i]+1,c,fabs((b-c)/c*100));
			if(j!=0)
			{if(fabs((b-c)/c*100)<error) break;}
			b=c;
		}
		else
		{
			printf("Root of section [%.0lf,%.0lf] is %lf and error
is %lf\n",x[i],x[i]+1,c,fabs((a-c)/c*100));
			if(j!=0)
			{if(fabs((a-c)/c*100)<error) break;}
			a=c;
		}
	}
	end=clock();
	printf("\n%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



2. 다음 방정식의 가장 작은 양의 근을 Newton법으로 구하고, 그 결과를 예제 2.6과 비교하라.

$$f(x) = e^{-x}(2\cos 2x + \sin 2x) = 0$$

단, 위 식의 도함수를 구할 때 다음 식을 이용하라.

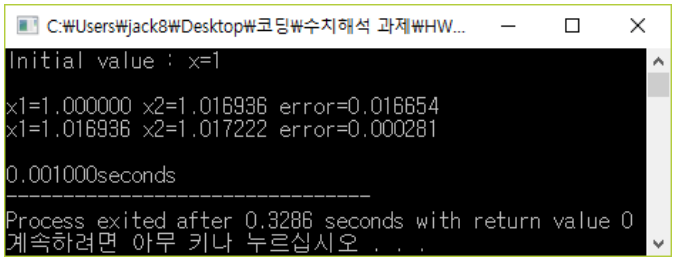
$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

여기서 h는 0.0001이다. 참값=1.0172

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return exp(-x)*(2*cos(2*x)+sin(2*x));
}
float diff(float y)
{
	return (func(y+0.0001)-func(y))/0.0001;
}
int main()
{
	//Incremental-Search Method
	int i,j;
	j=0;
	double x1,x2;
	for(i=0;i<=10;i++)
	{
		if(func(i)*func(i+1)<0)
		{
			x1=(double)i;
			j++;
			printf("Initial value : x=%.0lf\n\n",x1);
		}
		if(j==1) break;
	}
	double error=0.5/pow(10,5);
	//Newton Method
	clock_t start,end;
	start=clock();
	for(i=0;i<=100;i++)
	{
		x2=x1-func(x1)/diff(x1);
		printf("x1=%lf x2=%lf error=%lf\n",x1,x2,fabs((x2-x1)/x2));
		x1=x2;
		if(i!=0)
		{if(fabs((x2-x1)/x2)<error) break;}
	}
	end=clock();
	printf("\n%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



3. 다음 함수에서 주어진 구간 내에 있는 근을 5s까지 secant법으로 구하라.

(a) $\ln x = 0$, $0.01 < x < 2.0$ 참값=1.000

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
        return log(x);
}
int main()
{
        int i;
        double a=0.01, b=2.0;
        double c;
        double error=0.5/pow(10,5);
        clock_t start,end;
        start=clock();
        for(i=0;i<=100;i++)
                {
                        c=b-(b-a)/(func(b)-func(a))*func(b);
                        if(func(a)*func(c)<0)
                        {
                                printf("Root of section [%.3lf,%.3lf] is %lf and error
is %lf\n",a,b,c,fabs((b-c)/c*100));
                                if(i!=0)
                                {if(fabs((b-c)/c)<error) break;}
                                b=c;
                        }
                        else
                        {
                                printf("Root of section [%.3lf,%.3lf] is %lf and error
is %lf\n",a,b,c,fabs((a-c)/c*100));
                                if(i!=0)
                                {if(fabs((a-c)/c)<error) break;}
                                a=c;
                        }
                }
        printf("\n");
        end=clock();
        printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

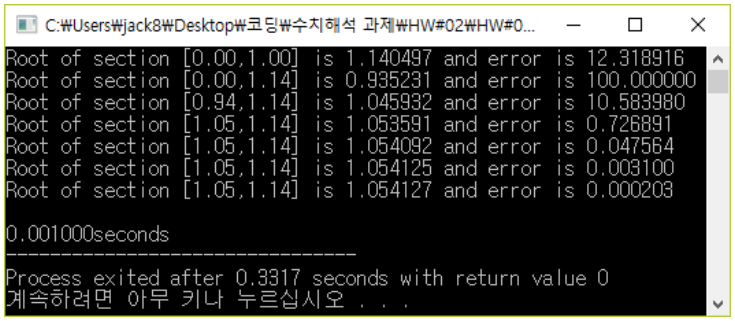
```
Root of section [0.010,2.000] is 1.739660 and error is 14.964980
Root of section [0.010,1.740] is 1.554019 and error is 11.945852
Root of section [0.010,1.554] is 1.419126 and error is 9.505365
Root of section [0.010,1.419] is 1.319584 and error is 7.543450
Root of section [0.010,1.320] is 1.245202 and error is 5.973492
Root of section [0.010,1.245] is 1.189056 and error is 4.721943
Root of section [0.010,1.189] is 1.146328 and error is 3.727304
Root of section [0.010,1.146] is 1.113602 and error is 2.938822
Root of section [0.010,1.114] is 1.088405 and error is 2.315025
Root of section [0.010,1.088] is 1.068926 and error is 1.822313
Root of section [0.010,1.069] is 1.053818 and error is 1.433640
Root of section [0.010,1.054] is 1.042070 and error is 1.127350
Root of section [0.010,1.042] is 1.032916 and error is 0.886180
Root of section [0.010,1.033] is 1.025773 and error is 0.696403
Root of section [0.010,1.026] is 1.020191 and error is 0.547143
Root of section [0.010,1.020] is 1.015825 and error is 0.429800
Root of section [0.010,1.016] is 1.012407 and error is 0.337575
Root of section [0.010,1.012] is 1.009730 and error is 0.265112
Root of section [0.010,1.010] is 1.007633 and error is 0.208184
Root of section [0.010,1.008] is 1.005988 and error is 0.163469
Root of section [0.010,1.006] is 1.004699 and error is 0.128352
Root of section [0.010,1.005] is 1.003687 and error is 0.100776
Root of section [0.010,1.004] is 1.002894 and error is 0.079121
Root of section [0.010,1.003] is 1.002271 and error is 0.062119
Root of section [0.010,1.002] is 1.001782 and error is 0.048768
Root of section [0.010,1.002] is 1.001399 and error is 0.038287
Root of section [0.010,1.001] is 1.001098 and error is 0.030055
Root of section [0.010,1.001] is 1.000862 and error is 0.023595
Root of section [0.010,1.001] is 1.000677 and error is 0.018523
Root of section [0.010,1.001] is 1.000531 and error is 0.014544
Root of section [0.010,1.001] is 1.000417 and error is 0.011416
Root of section [0.010,1.000] is 1.000327 and error is 0.008963
Root of section [0.010,1.000] is 1.000257 and error is 0.007036
Root of section [0.010,1.000] is 1.000202 and error is 0.005522
Root of section [0.010,1.000] is 1.000158 and error is 0.004336
Root of section [0.010,1.000] is 1.000124 and error is 0.003403
Root of section [0.010,1.000] is 1.000098 and error is 0.002673
Root of section [0.010,1.000] is 1.000077 and error is 0.002099
Root of section [0.010,1.000] is 1.000060 and error is 0.001648
Root of section [0.010,1.000] is 1.000047 and error is 0.001292
Root of section [0.010,1.000] is 1.000037 and error is 0.001015
Root of section [0.010,1.000] is 1.000029 and error is 0.000797
Root of section [0.010,1.000] is 1.000023 and error is 0.000625
Root of section [0.010,1.000] is 1.000018 and error is 0.000492

0.012000seconds
--------------------------------
Process exited after 0.3463 seconds with return value 0
```

(b) $e^x - \sin x - 2 = 0$, $0 < x < 1$ 참값=1.05413

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
        return exp(x)-sin(x)-2;
}
int main()
{
        int i;
        double a=0, b=1;
        double c;
        double error=0.5/pow(10,5);
        clock_t start,end;
        start=clock();
        for(i=0;i<=100;i++)
                {
                        c=b-(b-a)/(func(b)-func(a))*func(b);
                        if(func(a)*func(c)<0)
                        {
                                printf("Root of section [%.2lf,%.2lf] is %lf and error
is %lf\n",a,b,c,fabs((b-c)/c*100));
                                if(i!=0)
                                {if(fabs((b-c)/c)<error) break;}
                                b=c;
                        }
                        else
                        {
                                printf("Root of section [%.2lf,%.2lf] is %lf and error
is %lf\n",a,b,c,fabs((a-c)/c*100));
                                if(i!=0)
                                {if(fabs((a-c)/c)<error) break;}
                                a=c;
                        }
                }
        printf("\n");
        end=clock();
        printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

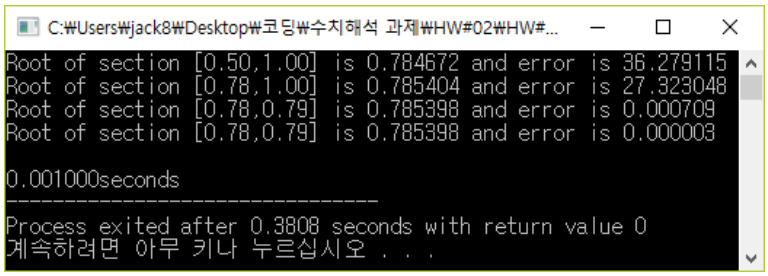


(c) $\sin x - \cos x = 0$, $0.5 < x < 1$ 참값=0.78540

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
        return sin(x)-cos(x);
}
int main()
{
        int i;
        double a=0.5, b=1;
        double c;
        double error=0.5/pow(10,5);
        clock_t start,end;
        start=clock();
        for(i=0;i<=100;i++)
                {
                        c=b-(b-a)/(func(b)-func(a))*func(b);
                        if(func(a)*func(c)<0)
                        {
                                printf("Root of section [%.2lf,%.2lf] is %lf and error
is %lf\n",a,b,c,fabs((b-c)/c*100));
                                if(i!=0)
                                {if(fabs((b-c)/c)<error) break;}
                                b=c;
                        }
                        else
                        {
                                printf("Root of section [%.2lf,%.2lf] is %lf and error
is %lf\n",a,b,c,fabs((a-c)/c*100));
                                if(i!=0)
                                {if(fabs((a-c)/c)<error) break;}
                                a=c;
                        }
                }
        printf("\n");
        end=clock();
        printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



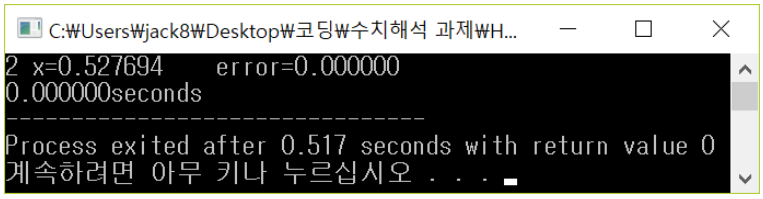
4. Muller법을 사용하여 다음 방정식에서 주어진 구간 내에 있는 근을 유효숫자 5자리까지 정확히 계산하라.

(a) $e^{-x} - 2x^2 = 0$, $1 < x < 2$(범위에 근이 없다) 참값=0.539835

<Code>

```
#include<stdio.h>
#include<math.h>
#include <time.h>
float func(float x)
{
	return exp(-x)-2*pow(x,2);
}
int main()
{
	clock_t start,end;
	start=clock();
	int i;
	double x0,x1,x2,h1,h2,g;
	x1=1;	x2=2;	x0=(x1+x2)/2;
	double x3;
	double error=0.5/pow(10,5);
	for(i=0;i<=100;i++)
	{
		h1=x0-x1;h2=x2-x0;g=h2/h1;
		double a,b,c;
		a=(g*func(x1)-func(x0)*(1+g)+func(x2))/(g*h1*h1*(1+g));
		b=(a*h1*h1+func(x0)-func(x1))/h1;
		c=func(x0);
		if (b<0)
		{
			x3=x0-(2*c)/(b-pow(b*b-4*a*c,0.5));

		}
		else
		{
			x3=x0-(2*c)/(b+pow(b*b-4*a*c,0.5));
		}
		if(x3>x0)
		{
			x0=x3;
			x1=x0;
			x2=x2;
			printf("1 x=%lf	error=%lf\n",x0,fabs((x0-x1)/x0));
			if(fabs((x0-x1)/x0)<error) break;
		}
		else
		{
			x0=x3;
			x1=x1;
			x2=x0;
			printf("2 x=%lf	error=%lf\n",x0,fabs((x0-x2)/x0));
			if(fabs((x0-x2)/x0)<error) break;
		}
	}
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

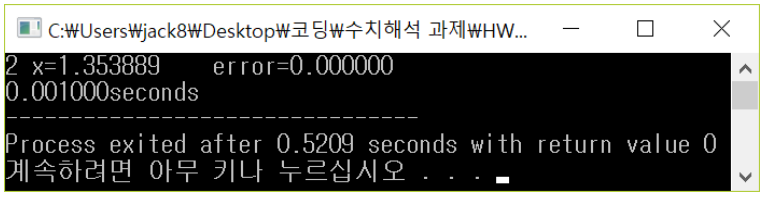


(b) $3x^2 + 4x^2 - 8x - 2 = 0$, $1 < x < 2$ 참값=1.3539

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return 3*x*x+4*x*x-8*x-2;
}
int main()
{
	int i;
	double x0,x1,x2,h1,h2,g;
	x1=1;	x2=2;	x0=(x1+x2)/2;
	double x3;
	double error=0.5/pow(10,5);
	clock_t start,end;
	start=clock();
	for(i=0;i<=100;i++)
	{
		h1=x0-x1;h2=x2-x0;g=h2/h1;
		double a,b,c;
		a=(g*func(x1)-func(x0)*(1+g)+func(x2))/(g*h1*h1*(1+g));
		b=(a*h1*h1+func(x0)-func(x1))/h1;
		c=func(x0);
		if (b<0)
		{
			x3=x0-(2*c)/(b-pow(b*b-4*a*c,0.5));

		}
		else
		{
			x3=x0-(2*c)/(b+pow(b*b-4*a*c,0.5));
		}
		if(x3>x0)
		{
			x0=x3;
			x1=x0;
			x2=x2;
			printf("1 x=%lf	error=%lf\n",x0,fabs((x0-x1)/x0));
			if(fabs((x0-x1)/x0)<error) break;
		}
		else
		{
			x0=x3;
			x1=x1;
			x2=x0;
			printf("2 x=%lf	error=%lf\n",x0,fabs((x0-x2)/x0));
			if(fabs((x0-x2)/x0)<error) break;
		}
	}
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

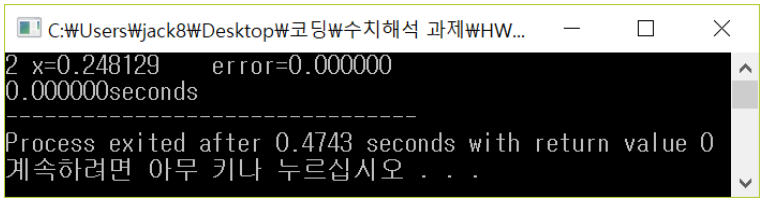


(c) $10e^{-x}\sin 2\pi x - 2 = 0$, $0.1 < x < 1$ 참값=0.449261

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return 10*exp(-x)*sin(2*M_PI*x)-2;
}
int main()
{
	int i;
	double x0,x1,x2,h1,h2,g;
	x1=0.1;  x2=1;    x0=(x1+x2)/2;
	double x3;
	double error=0.5/pow(10,5);
	clock_t start,end;
	start=clock();
	for(i=0;i<=100;i++)
	{
		h1=x0-x1;h2=x2-x0;g=h2/h1;
		double a,b,c;
		a=(g*func(x1)-func(x0)*(1+g)+func(x2))/(g*h1*h1*(1+g));
		b=(a*h1*h1+func(x0)-func(x1))/h1;
		c=func(x0);
		if (b<0)
		{
			x3=x0-(2*c)/(b-pow(b*b-4*a*c,0.5));

		}
		else
		{
			x3=x0-(2*c)/(b+pow(b*b-4*a*c,0.5));
		}
		if(x3>x0)
		{
			x0=x3;
			x1=x0;
			x2=x2;
			printf("1 x=%lf   error=%lf\n",x0,fabs((x0-x1)/x0));
			if(fabs((x0-x1)/x0)<error) break;
		}
		else
		{
			x0=x3;
			x1=x1;
			x2=x0;
			printf("2 x=%lf   error=%lf\n",x0,fabs((x0-x2)/x0));
			if(fabs((x0-x2)/x0)<error) break;
		}
	}
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

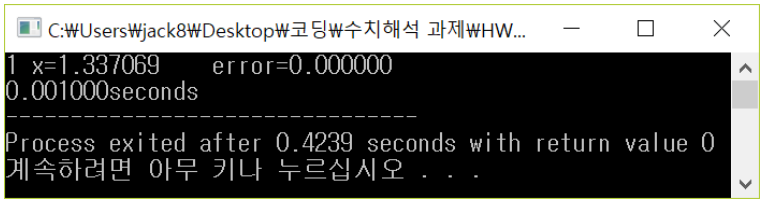


(d) $e^x - \sin\left(\frac{\pi x}{3}\right) - 3 = 0$, $0 < x < 2$ 참값=1.38447

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
float func(float x)
{
	return exp(x)-sin(M_PI*x/3)-3;
}
int main()
{
	int i;
	double x0,x1,x2,h1,h2,g;
	x1=0;	x2=2;	x0=(x1+x2)/2;
	double x3;
	double error=0.5/pow(10,5);
	clock_t start,end;
	start=clock();
	for(i=0;i<=100;i++)
	{
		h1=x0-x1;h2=x2-x0;g=h2/h1;
		double a,b,c;
		a=(g*func(x1)-func(x0)*(1+g)+func(x2))/(g*h1*h1*(1+g));
		b=(a*h1*h1+func(x0)-func(x1))/h1;
		c=func(x0);
		if (b<0)
		{
			x3=x0-(2*c)/(b-pow(b*b-4*a*c,0.5));

		}
		else
		{
			x3=x0-(2*c)/(b+pow(b*b-4*a*c,0.5));
		}
		if(x3>x0)
		{
			x0=x3;
			x1=x0;
			x2=x2;
			printf("1 x=%lf	error=%lf\n",x0,fabs((x0-x1)/x0));
			if(fabs((x0-x1)/x0)<error) break;
		}
		else
		{
			x0=x3;
			x1=x1;
			x2=x0;
			printf("2 x=%lf	error=%lf\n",x0,fabs((x0-x2)/x0));
			if(fabs((x0-x2)/x0)<error) break;
		}
	}
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



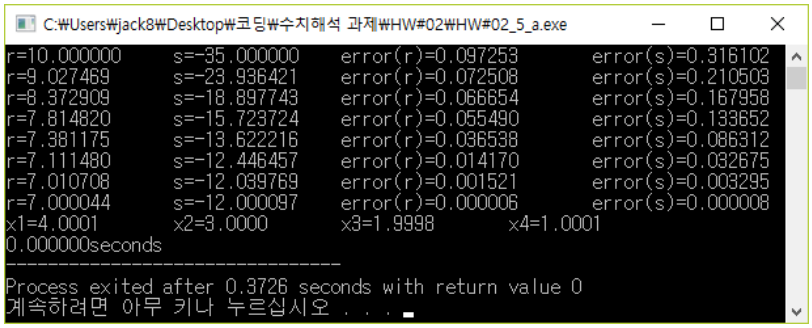
5. 다음 다항식의 근들을 Birstow방법을 이용하여 4s까지 계산하라.

(a) $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
int main()
{
	int i,j,k;
	j=4+2;  //order+2
	double A[5]={1,-10,35,-50,24}; //coefficient
	double r=-A[1]/A[0] , s=-A[2]/A[0]; //Larger root
	double B[7], C[7];
	B[0]=0,B[1]=0,C[0]=0,C[1]=0;
	double error=0.5/pow(10,4);
	clock_t start,end;
	start=clock();
	for(i=1;i<=50;i++)
	{
		for(k=2;k<=7;k++)
		{
			B[k]=A[k-2]+r*B[k-1]+s*B[k-2];
			C[k]=B[k]+r*C[k-1]+s*C[k-2];
			//printf("b%d=%lf  c%d=%lf\n",k-1,B[k],k-1,C[k]);
		}
		double dr,ds;
		dr=(B[j]*C[j-1]-B[j+1]*C[j-2])/(C[j]*C[j-2]-C[j-1]*C[j-1]);
		ds=(B[j+1]*C[j-1]-B[j]*C[j])/(C[j]*C[j-2]-C[j-1]*C[j-1]);
		printf("r=%lf     s=%lf    error(r)=%lf      error(s)=%lf\n",r,s,fabs(dr/r),fabs(ds/s));
		if(fabs(dr/r)<error&&fabs(ds/s)<error) break;
		r=r+dr;  s=s+ds;
	}
	double x1,x2,x3,x4;
	x1=(r+pow(r*r+4*s,0.5))/2;
	x2=(r-pow(r*r+4*s,0.5))/2;
	x3=(-B[3]+pow(B[3]*B[3]-4*B[2]*B[4],0.5))/(2*B[2]);
	x4=(-B[3]-pow(B[3]*B[3]-4*B[2]*B[4],0.5))/(2*B[2]);
	printf("x1=%.4lf  x2=%.4lf x3=%.4lf x4=%.4lf\n",x1,x2,x3,x4);
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



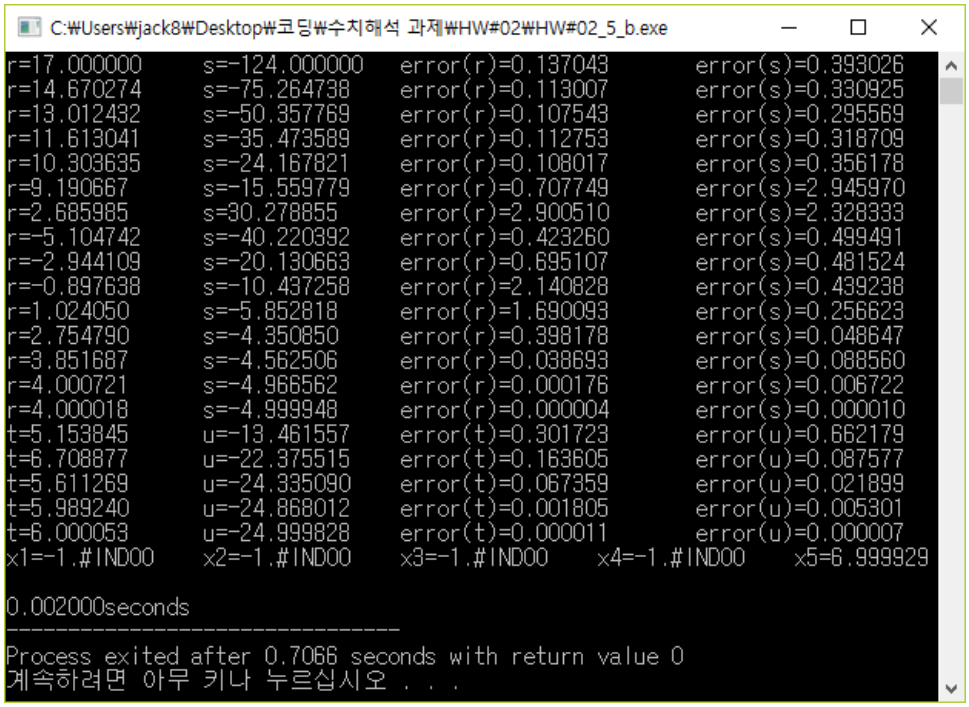
4s만 취하면 근은 4.000, 3.000, 2.000, 1.000이다.

(b) $x^5 - 17x^4 + 124x^3 - 508x^2 + 1035x - 875 = 0$ 참값=7.000

<Code>

```
#include<stdio.h>
#include<math.h>
#include<time.h>
int main()
{
	int i,j,k;
	j=5+1;  //order+1
	double A[6]={1,-17,124,-508,1035,-875}; //coefficient
	double r=-A[1]/A[0] , s=-A[2]/A[0]; //Larger root
	double B[8], C[8];
	B[0]=0,B[1]=0,C[0]=0,C[1]=0;
	double error=0.5/pow(10,4);
	clock_t start,end;
	start=clock();
	for(i=1;i<=50;i++)
	{
		for(k=2;k<=7;k++)
		{
			B[k]=A[k-2]+r*B[k-1]+s*B[k-2];
			C[k]=B[k]+r*C[k-1]+s*C[k-2];
		}
		double dr,ds;
		dr=(B[j]*C[j-1]-B[j+1]*C[j-2])/(C[j]*C[j-2]-C[j-1]*C[j-1]);
		ds=(B[j+1]*C[j-1]-B[j]*C[j])/(C[j]*C[j-2]-C[j-1]*C[j-1]);
		printf("r=%lf	s=%lf	error(r)=%lf
	error(s)=%lf\n",r,s,fabs(dr/r),fabs(ds/s));
		if(fabs(dr/r)<error&&fabs(ds/s)<error) break;
		r=r+dr;  s=s+ds;
	}
	double D[4]={B[2],B[3],B[4],B[5]};
	double E[6],F[6];
	E[0]=0, E[1]=0, F[0]=0, F[1]=0;
	double t=-D[2]/D[1] , u=-D[3]/D[1];//Smaller root
	for(i=1;i<=50;i++)
	{
		for(k=2;k<=5;k++)
		{
			E[k]=D[k-2]+t*E[k-1]+u*E[k-2];
			F[k]=E[k]+t*F[k-1]+u*F[k-2];
		}
		double dt,du;
		dt=(E[j-2]*F[j-3]-E[j-1]*F[j-4])/(F[j-2]*F[j-4]-F[j-3]*F[j-3]);
		du=(E[j-1]*F[j-3]-E[j-2]*F[j-2])/(F[j-2]*F[j-4]-F[j-3]*F[j-3]);
		printf("t=%lf	u=%lf	error(t)=%lf
	error(u)=%lf\n",t,u,fabs(dt/t),fabs(du/u));
		if(fabs(dt/t)<error&&fabs(du/u)<error) break;
		t=t+dt; u=u+du;
	}
	double x1,x2,x3,x4,x5;
	x1=(r+pow(r*r+4*s,0.5))/2;
	x2=(r-pow(r*r+4*s,0.5))/2;
	x3=(t+pow(t*t+4*u,0.5))/2;
	x4=(t-pow(t*t+4*u,0.5))/2;
	x5=-E[3]/E[2];
	printf("x1=%lf	x2=%lf	x3=%lf	x4=%lf	x5=%lf\n",x1,x2,x3,x4,x5);
	end=clock();
	printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>



x1, x2, x3, x4는 허근이고 x5는 4s로 나타내면 7.000이다.

(c) $16x^5 - 20x^3 + x^2 + 5x - 0.5 = 0$ 참값=-0.9758, -0.6086, 0.1021, 0.5564, 0.9260

<Code>

```c
#include<stdio.h>
#include<math.h>
#include<time.h>
int main()
{
	int i,j,k;
	j=5+1;  //order+2
	double A[6]={16,0,-20,1,5,-0.5}; //coefficient
	double r=-A[1]/A[0] , s=-A[2]/A[0]; //Larger root
	double B[8], C[8];
	B[0]=0,B[1]=0,C[0]=0,C[1]=0;
	double error=0.5/pow(10,4);
	clock_t start,end;
	start=clock();
	for(i=1;i<=50;i++)
	{
		for(k=2;k<=7;k++)
		{
			B[k]=A[k-2]+r*B[k-1]+s*B[k-2];
			C[k]=B[k]+r*C[k-1]+s*C[k-2];
		}
		double dr,ds;
		dr=(B[j]*C[j-1]-B[j+1]*C[j-2])/(C[j]*C[j-2]-C[j-1]*C[j-1]);
		ds=(B[j+1]*C[j-1]-B[j]*C[j])/(C[j]*C[j-2]-C[j-1]*C[j-1]);
		printf("r=%lf	s=%lf	error(r)=%lf	error(s)=%lf\n",r,s,fabs(dr/r),fabs(ds/s));
		if(fabs(dr/r)<error&&fabs(ds/s)<error) break;
		r=r+dr;  s=s+ds;
	}
```

```
        double D[4]={B[2],B[3],B[4],B[5]};
        double E[6],F[6];
        E[0]=0, E[1]=0, F[0]=0, F[1]=0;
        double t=-D[2]/D[1] , u=-D[3]/D[1];//Smaller root
        for(i=1;i<=50;i++)
        {
                for(k=2;k<=5;k++)
                {
                        E[k]=D[k-2]+t*E[k-1]+u*E[k-2];
                        F[k]=E[k]+t*F[k-1]+u*F[k-2];
                }
                double dt,du;
                dt=(E[j-2]*F[j-3]-E[j-1]*F[j-4])/(F[j-2]*F[j-4]-F[j-3]*F[j-3]);
                du=(E[j-1]*F[j-3]-E[j-2]*F[j-2])/(F[j-2]*F[j-4]-F[j-3]*F[j-3]);
                printf("t=%lf     u=%lf    error(t)=%lf
        error(u)=%lf\n",t,u,fabs(dt/t),fabs(du/u));
                if(fabs(dt/t)<error&&fabs(du/u)<error) break;
                t=t+dt; u=u+du;
        }
        double x1,x2,x3,x4,x5;
        x1=(r+pow(r*r+4*s,0.5))/2;
        x2=(r-pow(r*r+4*s,0.5))/2;
        x3=(t+pow(t*t+4*u,0.5))/2;
        x4=(t-pow(t*t+4*u,0.5))/2;
        x5=-E[3]/E[2];
        printf("x1=%lf    x2=%lf   x3=%lf   x4=%lf   x5=%lf\n",x1,x2,x3,x4,x5);
        end=clock();
        printf("%lfseconds",(double)(end-start)/CLOCKS_PER_SEC);
}
```

<Result>

```
C:₩Users₩jack8₩Desktop₩코딩₩수치해석 과제₩HW#02₩HW#02_5_c.exe
r=-0.000000     s=1.250000      error(r)=1.#INF00       error(s)=0.199332
r=-0.016694     s=1.000835      error(r)=1.275221       error(s)=0.082970
r=-0.037984     s=0.917796      error(r)=0.290050       error(s)=0.014727
r=-0.049001     s=0.904279      error(r)=0.017895       error(s)=0.000780
r=-0.049878     s=0.903574      error(r)=0.000070       error(s)=0.000003
r=-0.049881     s=0.903571      error(r)=0.000000       error(s)=0.000000
t=-6.895204     u=0.693348      error(t)=1.014862       error(u)=70.112990
t=0.102474      u=49.306048     error(t)=0.005203       error(u)=0.993132
t=0.101941      u=0.338634      error(t)=1.549226       error(u)=0.071821
t=-0.055989     u=0.362955      error(t)=0.057826       error(u)=0.067275
t=-0.052751     u=0.338537      error(t)=0.009328       error(u)=0.000195
t=-0.052259     u=0.338603      error(t)=0.000003       error(u)=0.000001
x1=0.925950     x2=-0.975831    x3=0.556353     x4=-0.608612    x5=0.102140
0.002000seconds
--------------------------------
Process exited after 0.3837 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

구한 근을 4s로 나타내면 0.9260, -0.9758, 0.5564, -0.6086, 0.1021이다.