Root Finding

Topic Coverage

- Assignment and expressions
- Nested control statements
- Value-returning functions

Problem Description

Numerical analysis is an important area in engineering. One simple numerical method for finding the root of a continuous function is the Bisection method. The **root**, `r`, of a continuous function `f` is a value such that `f(r) = 0`.

Given this polynomial function

$$p(x) = x^3 + 2x^2 + 5$$

we need to first provide two **endpoints** `a` and `b` such that `p(a)` and `p(b)` have opposite signs.

For example, let `a = -3` (hence `p(a) = -4`) and `b = 0` (hence `p(b) = 5`). Notice that `p(a)` and `p(b)` have opposite signs.

The principle is that, the root of the polynomial (the value `r` where `p(r) = 0`) must lie somewhere between `a` and `b`. So for the above polynomial, the root `r` must lie somewhere between `-3` and `0`, because `p(-3)` and `p(0)` have opposite signs.

The bisection method finds the midpoint `m` of the two endpoints `a` and `b`, and depending on the sign of `p(m)` (the function value at `m`), it replaces either `a` or `b` with `m` (so that `m` now becomes one of the two endpoints). It repeats this process and stops when either

- the midpoint `m` is the root, or
- the difference between the two endpoints `a` and `b` falls within a threshold `t`, that is, when they become very close to each other, i.e. `|a - b| < t`. Then the midpoint `m` is calculated as `(a + b)/2`, and this is the approximated root.
  We shall set the threshold to **0.0001** for this exercise.

The figure below shows the two endpoints `a = -3` and `b = 0`, their midpoint `m = -1.5`, and the function values at these three points: `p(a) = -4, p(b) = 5, p(m) = 6.125`.

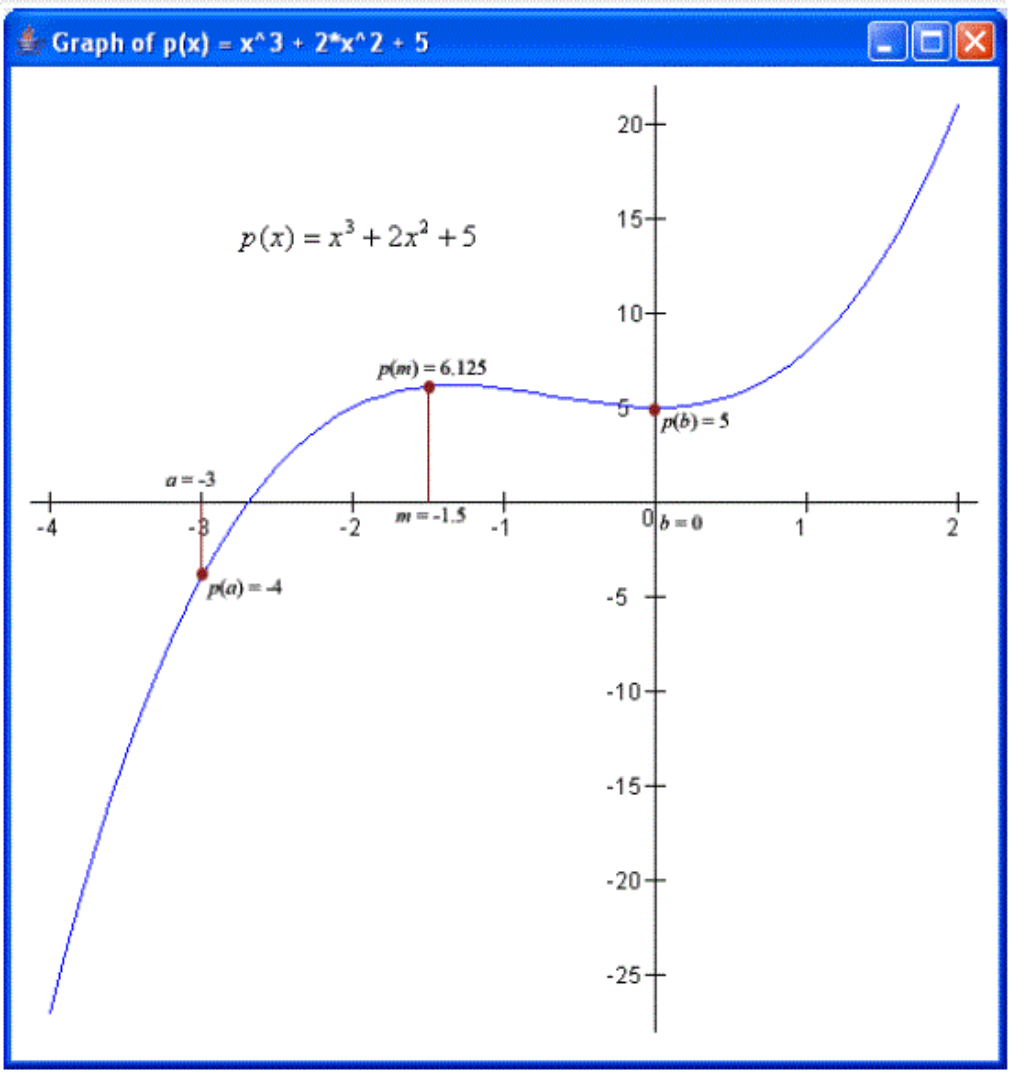

Figure. Graph of $p(x) = x^3 + 2x^2 + 5$

Since `p(m)` has the same sign as `p(b)` (both values are positive), this means that `m` will replace `b` in the next iteration.

The table below illustrates the iterations in the process. The end-point that is replaced by the mid-point value computed in the previous iteration is highlighted in pink background.

Iteration	endpoint *a*	endpoint *b*	midpoint *m*	Function value *p*(*a*)	Function value *p*(*b*)	Function value *p*(*m*)
1	-3.000000	0.000000	-1.500000	-4.000000	5.000000	6.125000
2	-3.000000	-1.500000	-2.250000	-4.000000	6.125000	3.734375
3	-3.000000	-2.250000	-2.625000	-4.000000	3.734375	0.693359
4	-3.000000	-2.625000	-2.812500	-4.000000	0.693359	-1.427002
5	-2.812500	-2.625000	-2.718750	-1.427002	0.693359	-0.312714
6	-2.718750	-2.625000	-2.671875	-0.312714	0.693359	0.203541
7	-2.718750	-2.671875	-2.695312	-0.312714	0.203541	-0.051243
8	-2.695312	-2.671875	-2.683594	-0.051243	0.203541	0.076980
9	-2.695312	-2.683594	-2.689453	-0.051243	0.076980	0.013077
10	-2.695312	-2.689453	-2.692383	-0.051243	0.013077	-0.019031
11	-2.692383	-2.689453	-2.690918	-0.019031	0.013077	-0.002964
12	-2.690918	-2.689453	-2.690186	-0.002964	0.013077	0.005059
13	-2.690918	-2.690186	-2.690552	-0.002964	0.005059	0.001048
14	-2.690918	-2.690552	-2.690735	-0.002964	0.001048	-0.000958
15	-2.690735	-2.690552	-2.690643	-0.000958	0.001048	0.000045
16	-2.690735	-2.690643	-2.690689	-0.000958	0.000045	-0.000456

Difference between *a* and *b* is < 0.0001

Hence the root of the above polynomial is **`-2.690689`** (because the difference between `a` and `b` in the last iteration is smaller than the threshold `0.0001`), and the function value at that midpoint `m` is **`-0.000456`**, close enough to zero.

Task

Write a program that asks the user to enter the integer coefficients (c_3, c_2, c_1, c_0) for a polynomial of degree 3:

$$\mathbf{c_3x^3 + c_2x^2 + c_1x + c_0}.$$

It then asks for the two endpoints, which are real numbers.

Take note of the following:

- Assume that the user enters a continuous function that has a real root.
- Assume that the two endpoints the user entered have function values that are opposite in signs.
- Assume that there is exactly one real root between the two endpoints.
- Use the **double** data types for real numbers.
- You may use the `pow` and `fabs` C Math library functions.

This task is divided into several levels. Read through all the levels (from first to last, then from last to first) to see how the different levels are related. **You may start from any level.**

- **Deadline: Submit your work to CodeCrunch by Thursday, 6 October, 23:59:59.**

Level 1

Name your program `root1.c`

Write a program that reads in four integer coefficients (`c3`, `c2`, `c1`, `c0`) followed by two floating-point values `a` and `b` representing the endpoints of the interval `[a,b]`. Output the values of the coefficients and endpoints on separate lines.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-3.0 0.0
1 2 0 5
a = -3.000000; b = 0.000000
```

Check the correctness of the output by typing the following Unix command

```
./a.out < root.in | diff - root1.out
```

To proceed to the next level (say level 2), copy your program by typing the Unix command

```
cp root1.c root2.c
```

Level 2

Name your program `root2.c`

Write a program that reads in four integer coefficients (`c3`, `c2`, `c1`, `c0`) followed by two floating-point values `a` and `b` representing the endpoints of the interval `[a,b]`. Output the values of the endpoints, as well as the values of the polynomial function evaluated at the two endpoints `p(a)` and `p(b)`.

You are encouraged to define the following function:

```
double polynomial(double x, int c3, int c2, int c1, int c0);
```

to compute the polynomial function value.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-3.0 0.0
a = -3.000000; b = 0.000000; p(a) = -4.000000; p(b) = 5.000000
```

Check the correctness of the output by typing the following Unix command

```
./a.out < root.in | diff - root2.out
```

To proceed to the next level (say level 3), copy your program by typing the Unix command

```
cp root2.c root3.c
```

Level 3

Name your program `root3.c`

Write a program that reads in four integer coefficients (`c3`, `c2`, `c1`, `c0`) followed by two floating-point values `a` and `b` representing the endpoints of the interval `[a,b]`. Output the values of the endpoints and the mid-point `m` between them, as well as the values of the polynomial function evaluated at these points `p(a)`, `p(b)` and `p(m)`.

You are encouraged to define the following function:

```
double polynomial(double x, int c3, int c2, int c1, int c0);
```

to compute the polynomial function value.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-3.0 0.0
a = -3.000000; b = 0.000000; m = -1.500000; p(a) = -4.000000; p(b) = 5.000000; p(m) = 6.125000
```

Check the correctness of the output by typing the following Unix command

```
./a.out < root.in | diff - root3.out
```

To proceed to the next level (say level 4), copy your program by typing the Unix command

```
cp root3.c root4.c
```

Level 4

Name your program `root4.c`

Write a program that reads in four integer coefficients (`c3`, `c2`, `c1`, `c0`) followed by two floating-point values `a` and `b` representing the endpoints of the interval `[a,b]`. Output the values of the endpoints and the mid-point `m` between them, as well as the values of the polynomial function evaluated at the two endpoints `p(a)`, `p(b)` and `p(m)`.

You are encouraged to define the following function:

```
double polynomial(double x, int c3, int c2, int c1, int c0);
```

to compute the polynomial function value.

Proceed to check if the root has been found using the current values of `a`, `b` and `p(m)` (refer to the two conditions given in the problem description). If the root is found, output the values of the root as well as the polynomial evaluated at the root. Otherwise output "`root not found`".

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-3.0 0.0
a = -3.000000; b = 0.000000; m = -1.500000; p(a) = -4.000000; p(b) = 5.000000; p(m) = 6.125000
root not found
```

```
$ ./a.out
1 0 0 0
-1.0 1.0
a = -1.000000; b = 1.000000; m = 0.000000; p(a) = -1.000000; p(b) = 1.000000; p(m) = 0.000000
root = 0.000000; p(root) = 0.000000
```

```
$ ./a.out
1 2 0 5
-2.690735 -2.690643
a = -2.690735; b = -2.690643; m = -2.690689; p(a) = -0.000959; p(b) = 0.000049; p(m) = -0.000455
root = -2.690689; p(root) = -0.000455
```

Check the correctness of the output by typing the following Unix command

```
./a.out < root.in | diff - root4.out
```

To proceed to the next level (say level 5), copy your program by typing the Unix command

```
cp root4.c root5.c
```

Level 5

Name your program `root5.c`

Write a program that reads in four integer coefficients (`c3`, `c2`, `c1`, `c0`) followed by two floating-point values `a` and `b` representing the endpoints of the interval `[a,b]`. Output the values of the endpoints and the mid-point `m` between them, as well as the value of the polynomial function evaluated at the two endpoints `p(a)`, `p(b)` and `p(m)`.

You are encouraged to define the following function:

```
double polynomial(double x, int c3, int c2, int c1, int c0);
```

to compute the polynomial function value.

Proceed to check if the root has been found using the current values of `a`, `b` and `p(m)` (refer to the two conditions given in the problem description). If the root is found, output the values of the root as well as the polynomial evaluated at the root.

If the root is not found, output "`root not found`", then make adjustments to the end-points of `a` and `b` as described in the problem statement above. Output the new end-point values, as well as their polynomial evaluations.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-3.0 0.0
a = -3.000000; b = 0.000000; m = -1.500000; p(a) = -4.000000; p(b) = 5.000000; p(m) = 6.125000
root not found
a = -3.000000; p(a) = -4.000000
b = -1.500000; p(b) = 6.125000
```

```
$ ./a.out
-1 2 0 5
0.0 3.0
a = 0.000000; b = 3.000000; m = 1.500000; p(a) = 5.000000; p(b) = -4.000000; p(m) = 6.125000
root not found
a = 1.500000; p(a) = 6.125000
b = 3.000000; p(b) = -4.000000
```

```
$ ./a.out
1 0 0 0
-1.0 1.0
a = -1.000000; b = 1.000000; m = 0.000000; p(a) = -1.000000; p(b) = 1.000000; p(m) = 0.000000
root = 0.000000; p(root) = 0.000000
```

```
$ ./a.out
1 2 0 5
-2.690735 -2.690643
a = -2.690735; b = -2.690643; m = -2.690689; p(a) = -0.000959; p(b) = 0.000049; p(m) = -0.000455
root = -2.690689; p(root) = -0.000455
```

Check the correctness of the output by typing the following Unix command

```
./a.out < root.in | diff - root5.out
```

To proceed to the next level (say level 6), copy your program by typing the Unix command

```
cp root5.c root6.c
```

Level 6

Name your program `root6.c`

Write a program that reads in four integer coefficients (`c3`, `c2`, `c1`, `c0`) followed by two floating-point values `a` and `b` representing the endpoints of the interval `[a,b]`.

For each iteration of the Bisection method, output the values of the endpoints and the mid-point `m` between them, as well as the value of the polynomial function evaluated at the two endpoints `p(a)`, `p(b)` and `p(m)`. At the end of the Bisection method, output the root as well as the value of the polynomial evaluated at the root on separate lines.

You are encouraged to define the following function:

```
double polynomial(double x, int c3, int c2, int c1, int c0);
```

to compute the polynomial function value.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-3.0 0.0
a = -3.000000; b = 0.000000; m = -1.500000; p(a) = -4.000000; p(b) = 5.000000; p(m) = 6.125000
a = -3.000000; b = -1.500000; m = -2.250000; p(a) = -4.000000; p(b) = 6.125000; p(m) = 3.734375
a = -3.000000; b = -2.250000; m = -2.625000; p(a) = -4.000000; p(b) = 3.734375; p(m) = 0.693359
a = -3.000000; b = -2.625000; m = -2.812500; p(a) = -4.000000; p(b) = 0.693359; p(m) = -1.427002
a = -2.812500; b = -2.625000; m = -2.718750; p(a) = -1.427002; p(b) = 0.693359; p(m) = -0.312714
a = -2.718750; b = -2.625000; m = -2.671875; p(a) = -0.312714; p(b) = 0.693359; p(m) = 0.203541
a = -2.718750; b = -2.671875; m = -2.695312; p(a) = -0.312714; p(b) = 0.203541; p(m) = -0.051243
a = -2.695312; b = -2.671875; m = -2.683594; p(a) = -0.051243; p(b) = 0.203541; p(m) = 0.076980
a = -2.695312; b = -2.683594; m = -2.689453; p(a) = -0.051243; p(b) = 0.076980; p(m) = 0.013077
a = -2.695312; b = -2.689453; m = -2.692383; p(a) = -0.051243; p(b) = 0.013077; p(m) = -0.019031
a = -2.692383; b = -2.689453; m = -2.690918; p(a) = -0.019031; p(b) = 0.013077; p(m) = -0.002964
a = -2.690918; b = -2.689453; m = -2.690186; p(a) = -0.002964; p(b) = 0.013077; p(m) = 0.005059
a = -2.690918; b = -2.690186; m = -2.690552; p(a) = -0.002964; p(b) = 0.005059; p(m) = 0.001048
a = -2.690918; b = -2.690552; m = -2.690735; p(a) = -0.002964; p(b) = 0.001048; p(m) = -0.000958
a = -2.690735; b = -2.690552; m = -2.690643; p(a) = -0.000958; p(b) = 0.001048; p(m) = 0.000045
a = -2.690735; b = -2.690643; m = -2.690689; p(a) = -0.000958; p(b) = 0.000045; p(m) = -0.000456
root = -2.690689
p(root) = -0.000456
```

Check the correctness of the output by typing the following Unix command

```
./a.out < root.in | diff - root6.out
```