

APS 105 – Computer Fundamentals

Lab #5: Recursion

Fall 1999

(To be completed *before* your lab period, week of October 11. Due to Thanksgiving on Monday, those students will have a makeup lab Tuesday 6–8pm.)

Objective: To understand how recursion works.

1 Printing A List of Numbers in Reverse Order

Design, implement and test a Java program that reads a list of numbers and prints them in reverse order. The number 0 will end the list, and is not to be included in the output. Use recursion to reverse the list. Do not use an array or any other data structure that has not yet been discussed in class.

Hint: Design a method "reverseNumbers" that reads an integer and returns it if it's 0; otherwise the method calls itself to process the other integers in the list and then prints the integer.

Your code will look something like this:

```
public class Reverse {
    public static void reverseNumbers () {
        System.out.print ("Enter an integer: ");
        int i = Stdin.getInt ();
        // **** PUT YOUR CODE HERE ****
    }
    public static void main (String[] args) {
        System.out.println ("Enter a list of integers, ending with 0.");
        reverseNumbers ();
    }
}
```

Execution of the program should be similar to the example that follows. The input supplied by the user is shown in **bold**.

```
Enter a list of integers, ending with 0.
Enter an integer: 8
Enter an integer: -2
Enter an integer: 3
Enter an integer: 0
3
-2
8
```

1

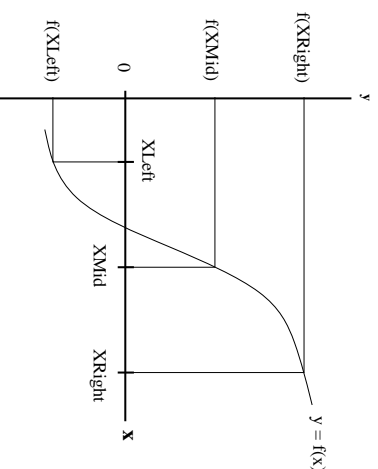
2 Finding Roots of Any Function

In lab assignment 3 you used an equation to determine the roots of a quadratic function. In general, it is more difficult to find roots of higher-order polynomials (those with high powers of x). In this section you will use a general method to directly search for the roots of a function, and you will use recursion to implement it. The method is called the "bisection method."

The bisection method is a means of finding an approximate root for the equation $f(x) = 0$ on the interval $xLeft$ to $xRight$ inclusive. (We will assume that the function is continuous in this interval.) You should ask the user for 3 double values:

- the interval endpoints, $xLeft$ and $xRight$
- the tolerance for the approximation, $epsilon$

The figure below shows an example situation:



The goal is to find an interval $[xLeft, xRight]$ that is less than $epsilon$ in length over which the function changes sign (and hence must be equal to zero, the root, somewhere between the two points). The midpoint of the interval

$$xMid = \frac{xLeft + xRight}{2.0}$$

will then be an approximation of the root of the equation.

If you happen to find a value of $f(xMid)$ that does equal zero then you are done and the program should stop. If the sign change happens in the interval $[xLeft, xMid]$ then let $xRight = xMid$, and similarly if the change is in the other interval. Then repeat the process.

2

3 Using Your Root Solver

In the sections below, you are to use your root solver program to find solutions for two fixed equations and one general equation.

For the first two parts, you will run your program with different values of $xLeft$, $xRight$, and $epsilon$ until you find the roots. If the $epsilon$ value you choose is too large, you may not find a root. If it is too small, your program may take a long time to find a root.

For Part C, you should prompt the user for the coefficients for a general cubic, as well as $xLeft$, $xRight$, and $epsilon$. Your program should try to find one root between $xLeft$ and $xRight$. Only Part C is going to be marked, but you should attempt to do Part A and B first to familiarize yourself with the concept before moving on to Part C.

For the optional Part D, your program should expand on part C by finding as many roots as it can. You may still ask the user for the values attempt to choose appropriate values between $xLeft$, $xRight$, and $epsilon$, or you can have your program attempt to choose them automatically.

Part A

Find an approximation to the equation

$$5x^3 - 2x^2 + 3 = 0$$

over the interval $[-1, 1]$ using $epsilon = 0.0001$. Your program should use a *recursive* implementation of the bisection method.

Part B

Modify your program from Part A to find approximations to the three roots of the equation

$$2x^3 - x^2 - 13x - 6 = 0$$

You may assume that all roots are real numbers.

Part C

Modify your solution to Part B to be a more general root solver. Ask the user for the coefficient values A , B , C , and D of the equation

$$A \cdot x^3 + B \cdot x^2 + C \cdot x + D = 0$$

You should also ask the user for $xLeft$, $xRight$, and $epsilon$. Your program should only find 1 root.

To speed up the user's search for the roots, you may find it helpful to repeatedly ask the user for new interval values. Call your bisection method for each new interval entered. If you do this, your program should end when an invalid interval is entered (for example, if $xLeft > xRight$).

Part D (OPTIONAL)

Instead of finding just a single root of the equation, modify the program so that it finds all of the roots of the equation, by cleverly dividing up and searching the space between $xLeft$ and $xRight$. You only have to find real roots, and your program must give up at some point (it must not run forever, nor should it run for an exceptionally long time). Also, instead of having the user enter $epsilon$ or the endpoints of the search interval, you may try to have your program determine them automatically.

Hint: Find the derivative and solve it using the quadratic equation. This will tell you how many local minima

and maxima the cubic equation has, as well as where they are. Use this knowledge to better select your search intervals.

4 Solution to Lab 3

Below you will find a sample solution to lab 3. Notice how variable names have been chosen to be descriptive. There are comments placed at strategic points to explain what is being done when it may not be completely clear. The corner cases have been carefully separated and handled appropriately (how many of you forgot to check for divide-by-zero? if you did, did you also check for an invalid equation when a and b are both zero but c is nonzero?).

```
// Quadratic Root Solver
// Sample Solution (Anthony Cox, Sept. 99)

// This program asks for coefficients to the equation
// a*x^2 + b*x + c = 0
// and then tries to find the solution(s) for x.
//
// It should handle all types of values for a, b, and c
// elegantly.

class Quad
{
    public static void main ( String[] args )
    {
        double a, b, c; // Coefficients
        double discriminant, // b^2-4ac part of formula
            root, // square root of discriminant
            twoA; // 2a part of formula

        // Whether this is centered depends on your window width...
        System.out.println (
            "
            Quadratic Equation Solver");
        System.out.println (
            "This program accepts three values for the user that":
            System.out.println (
            "correspond to the three variables in the quadratic equation":
            System.out.println (
            " ax^2 + bx + c = 0");
        System.out.println (
            "and solves for the roots of the equation.");

        // Initialization phase: get the first set of coefficients
        System.out.println (
            "Enter values -- all zeroes will terminate the program " );
        System.out.print ("Enter the value of a: ");
        a = StdIn.getDouble();
        System.out.print ("Enter the value of b: ");
        b = StdIn.getDouble();
        System.out.print ("Enter the value of c: ");
        c = StdIn.getDouble();
```

```

// Solve/print/input loop
while (((a == 0) && (b == 0) && (c == 0)))
{
    System.out.print ("The solutions of " + a + "x^2 + ");
    System.out.print (b + "x + ");
    System.out.println (c + " = 0 are");
    discriminant = b * b - 4 * a * c;
    twoA = 2 * a;

    // Find and print the roots
    if (a == 0) {
        // Avoid divide by zero error if a is zero.
        // There can be only 1 root.
        if (b == 0) {
            // here, a==0, b==0, c!=0, so the equation reads
            // "c = 0", which can't be true since c is nonzero.
            System.out.println ("Error: coefficients are not valid");
        } else {
            // we know b != 0, it's ok to divide now
            System.out.println ("Root 1: " + ((-1 * c) / b));
        }
    } else if (discriminant < 0) {
        // 2 complex roots
        root = Math.sqrt (-1 * discriminant);
        double realpart = -b / twoA;
        double imagpart = root / twoA;
        System.out.print ("Root 1: " + realpart + " + ");
        System.out.println (imagpart + "i");
        System.out.print ("Root 2: " + realpart + " - ");
        System.out.println (imagpart + "i");
    } else if (discriminant == 0) {
        // only 1 real root
        System.out.println ("Root 1: " + (-b / twoA));
    } else {
        // 2 real roots
        root = Math.sqrt (discriminant);
        System.out.println ("Root 1: " + ((-b + root) / twoA));
        System.out.println ("Root 2: " + ((-b - root) / twoA));
    }

    // Get the next set of coefficients
    System.out.println (); // blank line
    System.out.println ();
    *Enter values -- all zeroes will terminate the program *);
    System.out.print ("Enter the value of a: ");
    a = StdIn.getDouble ();
    System.out.print ("Enter the value of b: ");
    b = StdIn.getDouble ();
    System.out.print ("Enter the value of c: ");
    c = StdIn.getDouble ();
} // end while

// Termination
System.out.println ("End of Program");
}
}

```