

CSC 112 Lab 6: Recursion Examples

Fall 2009

Due: Sunday, October 11th, 9:00pm

Purpose

This lab is intended to:

- Allow one to gain experience in thinking recursively and in implementing recursive functions
- Reinforce basic looping and if-else constructs
- Reinforce file I/O and dynamic array allocation

Introduction

This lab will apply recursion to solve four different problems. When writing a recursive function, remember that you need to be thinking first in terms of *base case(s)* and then in terms of the *recursive step(s)*. **For each problem, clearly comment and explain each base case and recursive step in your code.**

Programming Task #1: Ternary Encoding

Write a program with a recursive function called `ternary` whose purpose is to generate and print the ternary equivalent of a positive integer. The ternary equivalent of a value x can be found by repeatedly examining the remainder digit of a value when divided by three, then recursively running the same process on the value divided by three. Your function should print the order of the digits correctly (left to right reading).

Examples:

```
ternary(5)      → 12
ternary(27)     → 1000
ternary(260)    → 100122
```

Design a main function to test your `ternary` function which has the following interface:

```
What value would you like to convert to ternary? user_input
answer
```

Save this program as `lab6_1.cpp`.

Programming Task #2: ArrayMax

Write a program with a recursive function called `max` that recursively finds the max of an integer array. The basic idea for your recursive function should be as follows:

- If you have a one element array, then that one element is the max of the array.
- Otherwise, find the max of the tail of the array (defined as all elements except the first element), and then compare the head against this max of the tail and return the max.

The program should be saved in a file called `lab6_2.cpp`. `main` should read in a file passed as a command line parameter. The file will be formatted as follows:

```
numberOfArrayEntries  
entry1 entry2 entry3 entry4 ...
```

For example, a file that appears as:

```
3  
2 1 3
```

would give an array of three elements, where `array[0] = 2`, `array[1] = 1`, `array[2] = 3`

An example run for this program would be:

```
./lab6_2 myArrayDescription.txt
```

Output should be written to the screen in the following format:

```
max = 3
```

Programming Task #3: Min-Max

Write a program with a recursive function `minmax` that finds the minimum and maximum values of an integer array. The function should take five parameters: the array, the first position in the array, the end position in the array, a pass-by-reference min value, and a pass-by-reference max value. The basic idea behind your recursive function should be as follows:

- If you have a one element array, then that one element is both the min and max of the array.
- If you have a two element array, then one comparison between the two values can be used. Based on that comparison, you can set the min and max appropriately.
- If you have more than two elements in the array, you should cut the array in half (as close as possible if you have an odd number of elements). Once this is done, you should find the min and max of each sub-array. The global minimum can then be found by comparing the two minimums returned from the sub-array calls and the global maximum can be found by comparing the two maximums returned from the sub-array calls.

The program should be saved in a file called `lab6_3.cpp`. `main` should read in a file passed as a command line parameter. The file will be formatted as follows:

```
numberOfArrayEntries
entry1 entry2 entry3 entry4 ...
```

For example, a file that appears as:

```
3
2 1 3
```

would give an array of three elements, where `array[0] = 2`, `array[1] = 1`, `array[2] = 3`

An example run for this program would be:

```
./lab6_3 myArrayDescription.txt
```

Output should be written to the screen in the following format:

```
min = 1
max = 3
```

Programming Task #4: List-Based Polynomials

Assume we want to represent polynomials with an array representation. Each entry in the array represents the coefficient for a corresponding power of x , with the first entry representing the constant (x^0) and successive entries representing the higher degrees. As an example, $x+3$ would be represented using `[3.0 | 1.0]` and x^3+4x-5 would be represented as `[-5.0 | 4.0 | 0.0 | 1.0]`.

Write a recursive function, called *polyeval*, to evaluate a polynomial for a given double input value. The first argument to *polyeval* should be the input polynomial, the second should be a real number for which evaluation is requested, and the third should be the size of the current polynomial array. Assume an empty input polynomial evaluates to 0 for any input x . You should base your function on Horner's Method for evaluating polynomials which says that any polynomial:

$$a_0 + a_1x^1 + a_2x^2 + \dots + a_nx^n$$

can be rewritten (factored) as

$$a_0 + x(a_1 + a_2x + \dots + a_nx^{n-1})$$

noting that starting at a_1 is another instance of a polynomial.

The program should be saved in a file called `lab6_4.cpp`. Your main function should ask the user for the input value x and read a file passed in as a command line parameter to load the polynomial, with that file having the following format:

```
maxDegree
coefficientN coefficientN-1 coefficientN-2 ... coefficient0
```

For example, for the polynomials described above the files would look like:

```
1
1.0 3.0
```

```
3
1.0 0.0 4.0 -5.0
```

Thus, running this program will be as follows:

```
./lab6_4 polynomialFile.txt
What value for X would you like to evaluate? user_input
answer_here
```

Completing the Lab

Please adhere to all of the following points to receive the maximum credit for this program:

- Turn-in electronically the four C++ source file for this program. Please tar and gzip your code into one `lab6.tar.gz` file. Submit this file under the “Lab 6” section of the Assignments Menu on Blackboard by the due date and time.
- Assume valid inputs.
- Use dynamic memory allocation for this program *where appropriate* for these programs. You are responsible for deleting any memory you dynamically allocate.
- Adhere to documentation and coding style standards
- As indicated in the Introduction section, please comment your base cases and recursive calls as to their purpose.