

CS 14: Data Structures and Algorithms

Lab 6: Binary Trees

Peter H. Fröhlich
phf@cs.ucr.edu

Wagner Truppel
wagner@cs.ucr.edu

Out on: February 10, 2003

Due by: February 16, 2003

Remember to follow the instructions and policies for assignments given on the course website. This week's assignment consists of **three** separate parts:

- an **in lab** programming practical (1)
- an **at home** programming exercise (2)
- an **at home** written exercise (3)

The programming practical is explained in more detail below. The “at home” parts are due **before 8:00 pm** on the date given above. You have to use the department's electronic turnin service for **everything**, including the **written** part. Put your solution for the latter into the **text file** `solution-6.txt` and turn it in together with the other files.

1 In-Lab: Programming Practical

A “programming practical” is an “applied (pop) quiz” testing your basic programming skills. You should think of this “practical” as an “exam” and not as a regular “lab” session: You are not allowed to talk, email, chat, or surf the web during the practical and should avoid **anything** that might be **perceived** as cheating. You will have **one hour** to work on your solution.

Your task for this “practical” will be to write a program that “draws a shape” made up of “*” characters, similar to the “Christmas Tree” lab from the first week of the quarter. Your program will **read one integer** from the user to determine the width and height of the shape. This integer must be **greater or equal to 5** and it must also be **odd**. Numbers **less than 5** as well as **even** numbers should be **rejected** with an error message.

The **exact shape** your program has to draw will be **announced by your TA** right before the practical starts. All we can tell you is that it will fit into a “square” of the dimensions given by the user. For example, if your shape was “square,” and the user entered “7” your program should print the following:

```
*****
*****
*****
*****
*****
*****
*****
```

Of course, the actual shape will be a **little** more complicated than that.

2 At-Home: Binary Trees

The archive for this week's lab contains a **skeleton** implementation of the binary tree data structure in the file `bintree-skeleton`. There's also a test driver `testbintree.cpp`.

Your task is to complete the implementation of class `BinaryTree` in a file `bintree`. As always, you should make things compile **before** you actually start working on your extensions (it's “Test early, test often!” all over again).

As requested, we have included more comments in the skeleton file for this lab. However, you will still have to add comments on your own to get a good “coding standard” score.

3 At-Home: A Tree of Problems

As discussed in lecture, there are a variety of ways for representing trees in terms of C++ constructs. For example, we can use “explicit `Node` objects” as in the programming exercise; we can represent trees as elements of an array; we can use parent pointers, child pointers, or both; and so on.

In this context, answer the following questions in **two or three sentences** each (the shorter the better):

1. The `parent()` function in `BinaryTree` requires $O(1)$ time: We just “follow a pointer” to find the parent of a given node. How would you implement `parent()` **without** maintaining a parent pointer in the `Node` class? What is the resulting worst-case complexity of `parent()`?
2. What representation has the **lowest** overhead—in terms of memory required **besides** the actual contents of each node—if we want to represent **perfect binary trees only**? Why?

Note: A binary tree is “perfect” if each level has **all** the nodes possible in that level (i.e. there are $2^h - 1$ nodes or 2^{h-1} leafs in a perfect binary tree of height h).

3. Can we represent an **arbitrary n-ary tree** in terms of a **binary tree**? How?

Hint: For this question, you should interpret “binary tree” to mean “two pointers per node” instead of “pointers to left and right child.”

Answer **in your own words** if you want to avoid being accused of cheating! If you use sources **other** than the assigned books for your answer, be sure to include a **reference** as well!