CS 122 Homework 4 Fall 2004

Due 11:00AM, Wednesday, November 10th, 2004

Name: ____________________________ Student ID: ____________________________

Instructions: There is a 20% deduction for late homework. The deduction becomes 50% if the homework is two days late. No credit is given after three days. You are to work individually, and all work should be your own. Come up with solutions to the following questions and then submit your answers in a word processed format (you may use whatever word processor you prefer) as an e-mail attachment. E-mail your file to aroth@nmt.edu with the subject line of "[CS 122] Homework 4" by the due-date posted on the website.

Grading Criteria:

Each question (except for 11) is worth 10 points.

If you answer questions 11, you may earn up to 15 bonus points.

Partial credit will be given for partially correct answers and/or incorrect answers that show that a good amount of effort was put forth in trying to arrive at the correct answer, as long as an explanation of the answer is provided (i.e. so if you are unsure, don't just write down your answer, right down how you arrived at it and why you think it is/isn't the correct answer).

Points will be deducted for late assignments.

Questions:

1. Consider stack *S*, which is initialized with values 17, 330, and 4, in that order (with 17 at the top of stack). What are the contents of *S* after the execution of the following code, where 'S.push()' pushes a new value onto the top of the stack, and 'S.pop()' removes the item from the top of the stack and returns its value:

```
int x = 7;
S.push(32);

while(x != 20)
{
        x = S.pop();
        x--;
        S.push(x);
}
```

```
S.push(x + S.pop() + S.pop());
```

2. Why is it inefficient to use a recursive function to compute Fibonacci numbers? In what other conditions may it be inappropriate to use a recursive solution?

   Recall that fibonacci(n) = fibonacci(n – 1) + fibonacci(n – 2)

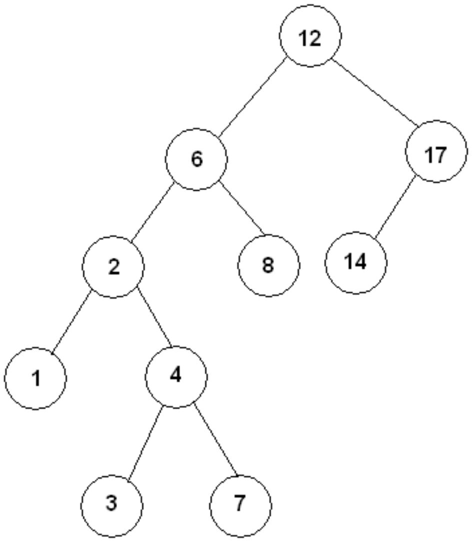
3. What are some of the most common applications of trees?

4. If you have a balanced tree in which every node has a given, fixed number of children, is it necessary to explicitly create a new structure in order to represent the tree in memory, or is it possible to represent the tree by using only built-in types (assume that you are using C++)? Why or why not?

5. Consider the case described in the above example, except in this case a node may have a variable number of children. Is it necessary to explicitly define a new structure to represent this tree? Why or why not?

6. What are the advantages of using a tree structure to store data instead of a linked list? What are the disadvantages?

7. Consider the tree pictured below, is it a valid binary search tree? Why or why not?



8. Why is deletion problematic when dealing with a properly sorted binary search tree? What happens in the case when a leaf node is deleted? A node with a single child? A node with two children?

9. What are the fundamental properties associated with a B-tree of order M? In what kind of situation would it be desirable to use a B-tree to store data?

10. What is the difference between a tree and a graph? Are all trees graphs? Are all graphs trees? If a graph is acyclic, is it necessarily equivalent to a tree?

11. (15 Bonus Points) Suppose that graph *G* has the following adjacency lists:

$1 - (2; 3; 4)$
$2 - (1; 3; 4)$
$3 - (1; 2; 4)$
$4 - (1; 2; 3; 6)$
$5 - (6; 7; 8)$
$6 - (4; 5; 7)$
$7 - (5; 6; 8)$
$8 - (5; 7)$

(1) (5 Bonus Points) Draw *G*.

(2) (5 Bonus Points) Give the sequence of vertices visited using depth-first search starting at
vertex 1.

(3) (5 Bonus Points) Give the sequence of vertices visited using breadth-first search starting at
vertex 1.