CSci 115 Fall 2000 Midterm 2
21 points
There are 12 problems and 9 pages
Show your work.

1. (1 point – no partial credit) Give the inorder traversal of the following tree:

2. (1 point) Calculate the external path length of the tree given in problem 1. Root is at level 1. Show your work.

3. (a) (1/2 point – no partial credit) Using the convention of Wirth (root is at level 1), what is the maximum number of nodes in a tree of degree d at level i?

   (b) (1/2 point – no partial credit) Using the convention of Wirth, what is the maximum number of nodes in a tree of degree d with height h?

4. (2 points) Huffman's Algorithm: You are given the following data structures declarations in Pascal. For C or C++ users, use similar declarations.

```
Const maxsize = 100; maxsym = 26;
Type
   treeelt = record
      leftchild, rightchild, parent : integer
      end;
   alphabetelt = record
      symbol : char;
      probability : real;
      Leaf : integer
   forestelt = record
      weight : real;
      root : integer
   end;
Var
   TREE : array [1..maxsize] of treeelt;
   ALPHABET : array [1..maxsym] of alphabetelt;
   FOREST : array [1..maxsym] of forestelt;

etc.
```

You are given some description of two subprograms in Pascal. For C and C++ users, use the same names for the subprograms, respectively.

```
procedure lightones ( var least, second : integer );
(* sets least and second to the indices in FOREST of the
   trees of smallest weight.  We assume lasttree >= 2. *)

function create ( lefttree, righttree : integer ) : integer;
(* returns new node whose left and right children are
   FOREST[lefttree], root, and FOREST[righttree].root *)
```

Write a procedure (in pseudocode) for the Huffman's algorithm.

5. (2 points) Construction of Huffman tree.

   Suppose characters a, b, c, d, e have probabilities 0.32, 0.13, 0.33, 0.08, 0.14, respectively.

   Find an optimal Huffman code (0.5 point).

   Draw the Huffman tree and detail its construction (1.25 point).

   Show your work.

   What is the average code length? (0.25 point)

6. (3 points) Backtracking Algorithm: Given an $n \times n$ board, declaration, function prototype, and a main function in C, code the backtracking algorithm for the try function for the knight's tour. If you code in C++, please make sure that your declaration, and main function are as close to the C version as possible.

```
\#include<stdio.h>
\#include<stdlib.h> /* in case you need it */

\#define n 8
\#define nsq 64
\#define TRUE 1
\#define FALSE 0

int i, j;
int *q;
int a[9], b[9]; /* arrays for the moves of the knight */
int h[n+1][n+1]; /* array for the board */

void try(int ith_move, int x_startingposition, int y_startingposition, int *q);
/* q is used to report on the success or failure of
   this move in achieving a complete solution */

main() {
   /* legal moves for the knight */
   a[1] = 2;  a[2] = 1;  a[3] = -1; a[4] = -2;
   a[5] = -2; a[6] = -1; a[7] = 1;  a[8] = 2;
   b[1] = 1;  b[2] = 2;  b[3] = 2;  b[4] = 1;
   b[5] = -1; b[6] = -2; b[7] = -2;  b[8] = -1;

   /* initialization of the board to indicate
      that none of the squares have been visited */
   for (i=1; i<=n; i++)
      for(j=1; j<=n; j++)
         h[i][j] = 0;

   h[1][1] = 1; /* initial position of the knight, the first move */
   try (2, 1, 1, &q);

   /* assume we have the code to print the result here */
}
```

7. (2 points – no partial credit)

   Insert J into the given AVL tree.
   Draw the resulting three with J inserted.
   Rebalance if necessary.

8. (2 points) Optimal Search Tree: Given the following set of frequencies:

internal node		x_1		x_2		x_3	
external node	y_0		y_1		y_2		y_3
frequencey	1	5	2	3	1	4	5

   Construct an optimal search tree for this data.
   Show all your work.

9. (2 points – no partial credit)

   Delete W from the given AVL tree.]
   Draw the resulting tree with W deleted.
   Rebalance if necessary.

10. (2 points) Fibonacci Tree Construction (using Wirth's convention with root is at level 1):

    A Fibonacci tree of height 0 is the empty tree.
    Construct a Fibonacci tree with height 5. (1.75 points)
    How many nodes are required? (0.25 point)

11. (1.5 points) Given n nodes, write a recursive algorithm (in pseudocode) to construct a perfectly balanced binary tree with n nodes.

12. (1.5 points) Self-assessment questions on Binary Search Trees. (0.5 point each)

    (a) A binary search tree is defined as
        i. a finite set of nodes which either is empty or consists of a root node with two disjoint binary trees.
        ii. a binary tree used for searching
        iii. (*) a binary tree such that for each node all keys in the left subtree of the node are less than the key in the node and those in the right subtree are greater than the key in the node.
        iv. a binary tree whose nodes contain keys arranged in descending order along every path from the root to a leaf.

    (b) An AVL tree is defined as
        i. a tree which has two subtrees whose heights differ by at most 1.
        ii. a binary tree whose leaves appear on at most two adjacent levels.
        iii. a binary tree whose longest paths in the left subtree and the right subtree differ by at most 1.
        iv. (*) a tree such that it and each of its subtrees have the property that the heights of the left and right subtrees of the root differ by at most 1.

    (c) Given n keys K_i, $i = 1, 2, \ldots, n$, and associated frequency counts of successful search a_i, $i = 1, 2, \ldots, n$, and unsuccessful search b_i, $i = 0, 1, \ldots, n$, we can build an optimal tree bottom-up based on the following main principle.
        i. Optimal trees have all their subtrees such that the root is the node of the highest weight in the subtree.
        ii. Optimal trees have all their subtrees such that the root balances the weights in the subtree.
        iii. (*) All subtrees of an optimal tree are optimal.
        iv. The weighted path length of the optimal tree is the sum of the weighted path lengths of right and left subtrees plus the weight.