Exam 3

Practice Exam

- Do not open this exam until we instruct you to do so.
- This is a closed book, closed notes, closed electronics, closed neighbor, pencil and paper exam.
- This exam is designed to be completed in 50 minutes. However, you may take 120 minutes to work on the questions. You must immediately turn in the exam when we call for it.
- Write answers in the spaces provided. Indicate your final answer clearly. Cleanly erase or cross out any work you do not want graded. Show your work and explain your reasoning where appropriate; this will help to earn partial credit.
- Backpacks must be left at the front of the room.
- Electronic devices of any kind, including cell phones and calculators, must be completely turned off and stowed in your backpack.
- The exam must be written with a pencil.

For grading:

Page 2 _____

Page 3 _____

Page 4 _____

Page 5 _____

Page 6 _____

Page 7 _____

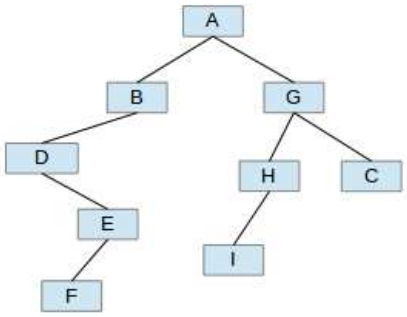
Total _____

Grade _____

1. Consider the following definitions:

```
typedef char T;
struct node {
  T data;
  node *left;
  node *right;
  node(const T & data,node *left = NULL, node *right = NULL)
    : data(data), left(left), right(right) {}
};
```

   (a) Give a single nested expression to build the tree shown below:



   (b) Assume a queue of `node` pointers. Using the tree from part a, tell what is output by the following fragment.

```
queue container;  // initially empty queue
container.add(tree);
while (!container.empty()) {
  node *p = container.front();
  container.remove();
  cout << p->data << endl;
  if (p->left)
    container.add(p->left);
  if (p->right)
    container.add(p->right);
}
```

2. Two trees:

   (a) Draw the binary search tree created by inserting the following integers into an initially empty tree, in the order given:

   ```
   14, 17, 14, 6, 8, 22, 16, -3, 5, 9
   ```

   (b) Draw a complete binary search tree containing only the integers:

   ```
   14, 12, 8, 20, 16, 7, 6, 2, 19
   ```

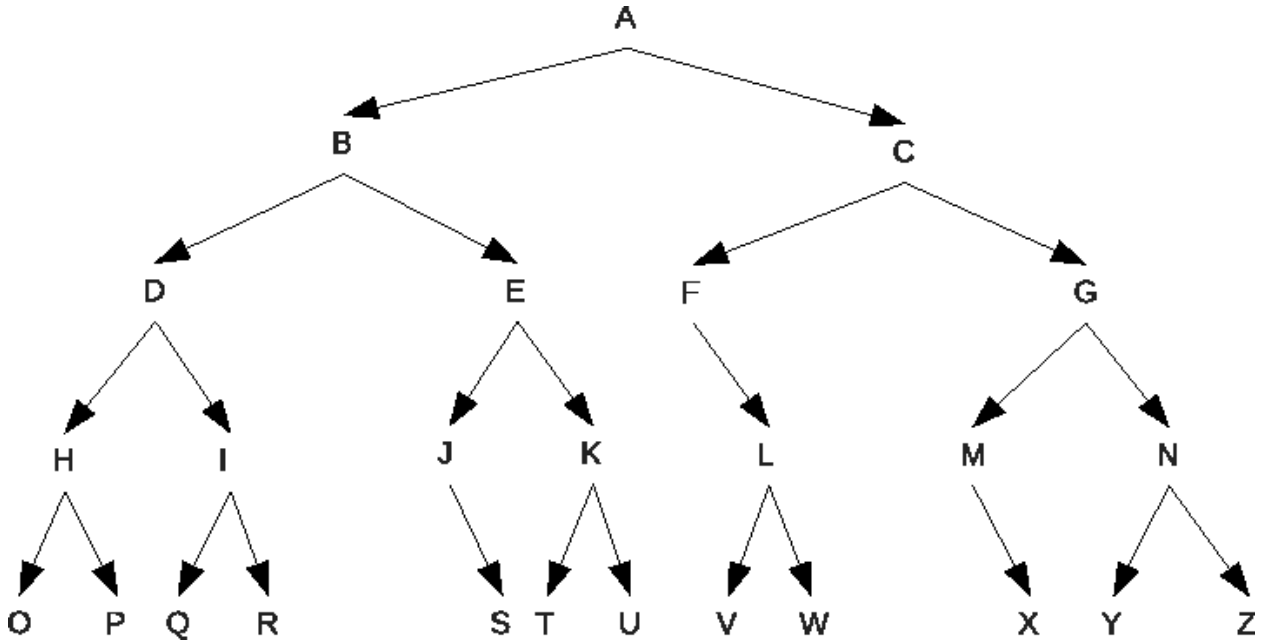
3. Returning to the definition of a binary tree node provided in question 1, write a recursive function that determines whether two such trees are equal. Trees are equal only if they have exactly the same structure with equal data elements in every corresponding pair of nodes.

```
bool tree_equal(node * t1, node * t2)
{




}
```

4. Consider this binary tree.



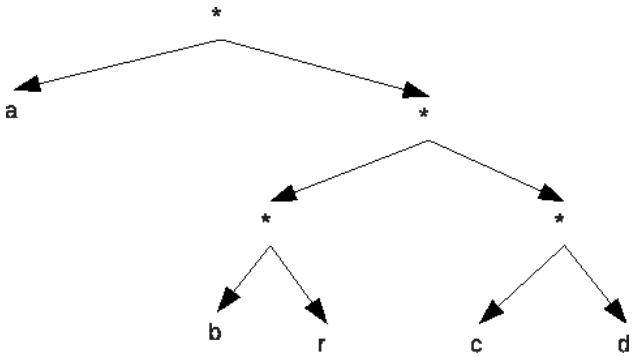
(a) What is the depth of node R?

(b) What is the height of the tree?

(c) List the nodes in the order they would be visited in a left-to-right postorder traversal.

(d) Calculate the branching factor of this tree, as a fraction.

5. Imagine a tree that encodes characters by placing them at the leaves of the tree. A character's code is the path of left and right branches from the root to the leaf. A path is a string of `'L'` and `'R'` characters. For example, in the tree below, the path `"RLL"` encodes the character `'b'`; the path `"L"` encodes `'a'`. The asterisks represent internal nodes. Internal nodes don't encode anything.



Write a function that takes a tree root and a path, and returns the character at the leaf. You may assume that the path will end at a leaf. Hint: Recall that the `substr(pos,n)` method returns a substring starting at position `pos` and having length `n`. For example, this outputs `DE`:

```
    string p = "ABCDE";
    cout << p.substr(3,2);
```

```
struct h_node {
  char data;
  h_node* left;
  h_node* right;
};

char decodePath(h_node * root, string path)
{









}
```