Program 3 due 10 pm Sunday, November 6th

Last Time
Recursion
- more practice writing recursive code
- complexity of recursive methods
- practice analyzing complexity

Today
Recursion
- more practice writing/analyzing recursion (from last time)
- execution tree tracing
Searching
Categorizing ADTs Part 1
General Trees
- implementing
- determining tree height

Next Time
Read: finish *Trees*, start *Binary Search Trees*
Binary Trees
- implementing
Tree Traversals
Categorizing ADTs Part 2
Binary Search Tree (BST)
- BSTnodes
- BST class
Picking Lottery Numbers

What are your odds of winning the lottery? It depends on the number of possible combinations given how many numbers you have to pick and over what range:

**Supercash** - choose 6 out of 39 numbers (range 1 – 39)

**Megabucks** - choose 6 out of 49 numbers (range 1 – 49)

N Choose K: How many unique combinations of K things can you make from N things?

Recursive Definition:

\[ c(n,k) = \]

Base case(s):

\[ \rightarrow \text{Implement the } c(n,k) \text{ method.} \]
Execution Tree Tracing of $c(n,k)$
Searching

Linear Search:

Binary Search:
Categorizing ADTs Part 1
General Tree

The Tree Node Class:

class Treenode<T> {
    private T data;
    private ListADT<Treenode<T>> children;
    ...
}

→ Draw a picture of the memory layout of a Treenode (assume an ArrayList is used for the ListADT):

The Tree Class:

public class Tree<T> {
    private Treenode<T> root;
    private int size;

    public Tree() {
        root = null;
        size = 0;
    }
    ...

→ Draw a picture of the memory layout of an empty general tree:

→ Draw a picture of the memory layout of a general tree with a root node having 3 children:
Determining Height of a General Tree

Recall the height of a tree is the length of a path from the root to the deepest leaf.

→ Write a recursive definition for the height of a general tree.

→ Complete the recursive height method based on the recursive definition. Assume the method is added to a Tree class having a root instance variable.

```java
public int height() {
```