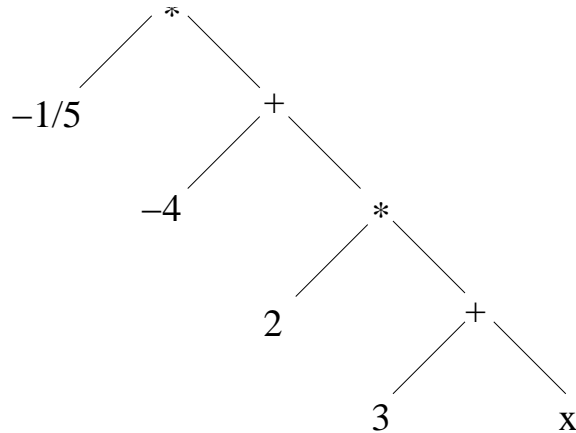


## Homework 4, CS620, McConnell, Spring '08, due 5/8

Modified 5/6 to give more hints.

1. Suppose you have an adjacency-list representation of a graph, where the nodes are numbered from 1 through  $n$ , the adjacency list of each node  $i$  is pointed to by  $A[i]$ , and the adjacency lists are sorted in ascending order of node number. Somebody rennumbers the nodes with a new permutation of 1 through  $n$ . Tell how, in  $O(n + m)$  time, you can construct an adjacency-list representation where, if a node's new number is  $j$ , its adjacency list is pointed to by  $A[j]$ , and each adjacency list is sorted in ascending order of the new node numbering.
2. Come up with a way to implement the algorithm of Problem 4 of the second midterm in  $O(n+m)$  time. *Hint: apply the result of Problem 1 to the numbering given by a perfect elimination order. Also, consider keeping a sorted list for each maximal clique, listing its members, and labeling each vertex with a pointer to one of the cliques it is a member of. Part of your solution should be a justification of the following claim: the sum of cardinalities of the maximal cliques is  $O(n + m)$ .*
3. Nobody got the extra credit from the last homework, so let me re-give it with some more hints.
  - (a) For the nested expression  $E(x) = (1/5 * (-4 + (2 * (3 + x))))$ , notice the shape of the parse tree:



We can implement this with a splay tree by having it use the list representation described in the last assignment. Each subtree of the splay tree represents a sublist, hence a smaller nested expression. Let us add the invariant that each internal node stores the short form for its subexpression. Show that this does not affect the running time of any of the splay-tree (path) operations.

4. Show how to solve the last problem from the last assignment by using a linking/cutting tree to represent the parse tree of the fully-parenthesized expression.

*Hint: Because the operators are commutative, the order of left and right children in a parse tree doesn't matter, so there's no need to change your splay-tree representation of a "wiggly" nested expression like this one:*

