In-Class Problems Week 5, Fri.

Problem 1.

Definition. The recursive data type binary-2PG of *binary trees* with leaf labels L is defined recursively as follows:

- **Base case:** $\langle \texttt{leaf}, l \rangle \in$ binary-2PG, for all labels $l \in L$.
- **Constructor case:** If $G_1, G_2 \in$ binary-2PG, then

$$\langle \texttt{bintree}, G_1, G_2 \rangle \in \text{binary-2PG}.$$

The *size* $|G|$ of $G \in$ binary-2PG is defined recursively on this definition by:

- **Base case:**

$$|\langle \texttt{leaf}, l \rangle| ::= 1, \quad \text{for all } l \in L.$$

- **Constructor case:**

$$|\langle \texttt{bintree}, G_1, G_2 \rangle| ::= |G_1| + |G_2| + 1.$$

For example, the size of the binary-2PG G pictured in Figure 1, is 7.

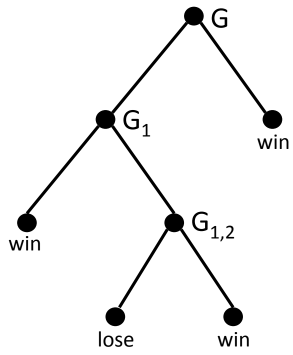

Figure 1 A picture of a binary tree G.

(a) Write out (using angle brackets and labels `bintree`, `leaf`, etc.) the binary-2PG G pictured in Figure 1.

The value of flatten(G) for $G \in$ binary-2PG is the sequence of labels in L of the leaves of G. For example, for the binary-2PG G pictured in Figure 1,

$$\text{flatten}(G) = (\texttt{win}, \texttt{lose}, \texttt{win}, \texttt{win}).$$

(b) Give a recursive definition of flatten. (You may use the operation of *concatenation* (append) of two sequences.)

(c) Prove by structural induction on the definitions of flatten and size that

$$2 \cdot \text{length}(\text{flatten}(G)) = |G| + 1. \tag{1}$$

Problem 2.
For $T \in$ BBTr, define

$$\begin{aligned}
\text{leaves}(T) &::= \{S \in \text{Subtrs}(T) \mid S \in \text{Leaves}\} \\
\text{internal}(T) &::= \{S \in \text{Subtrs}(T) \mid S \in \text{Branching}\}.
\end{aligned}$$

(a) Explain why it follows immediately from the definitions that if $T \in$ Branching,

$$\text{internal}(T) = \{T\} \cup \text{internal}(\text{left}(T)) \cup \text{internal}(\text{right}(T)), \tag{trnlT}$$

$$\text{leaves}(T) = \text{leaves}(\text{left}(T)) \cup \text{leaves}(\text{right}(T)). \tag{lvT}$$

(b) Prove by structural induction on the definition of RecTr that in a recursive tree, there is always one more leaf than there are internal subtrees:
Lemma. *If $T \in$ RecTr, then*

$$|\textit{leaves}(T)| = 1 + |\textit{internal}(T)|. \tag{lf-vs-in}$$

Problem 3.

Definition. Define the *sharing binary trees* SharTr recursively:

Base case: ($T \in$ Leaves). $T \in$ SharTr.

Constructor case: ($T \in$ Branching). If left(T), right$(T) \in$ SharTr, then T is in SharTr.

(a) Prove size (T) is finite for every $T \in$ SharTr.

(b) Give an example of a finite $T \in$ BBTr that has an infinite path.

(c) Prove that for all $T \in$ BBTr

$$T \in \text{SharTr} \longleftrightarrow T \text{ has no infinite path.}$$

(d) Give an example of a tree $T_3 \in$ BBTr with three branching subtrees and one leaf.

(e) Prove that
Lemma. *If $T \in$ SharTr, then*

$$|\textit{leaves}(T)| \leq 1 + |\textit{internal}(T)|.$$

Hint: Show that for every $T \in \text{SharTr}$, there is a recursive tree $R \in \text{RecTr}$ with the same number of internal subtrees and at least as many leaves.

Problem 4. (a) Edit the labels in this size 15 tree T so it becomes a search tree for the set of labels $[1..15]$.

```
             T
            / \
           /   \
          /     \
         /       \
       A           B
      / \         / \
     /   \       /   \
    /     \     /     \
   C       D   E       F
  / \     / \         / \
 /   \   /   \       /   \
G     H I     J     K     L
             / \
            M   N
```

(b) For any recursive tree and set of labels, there is only one way to assign labels to make the tree a search tree. More precisely, let num : RecTr $\to \mathbb{R}$ be a labelling function on the recursive binary trees, and suppose T is a search tree under this labelling. Suppose that num_{alt} is another labelling and that T is also a search tree under num_{alt} for the *same* set of labels. Prove by structural induction on the definition of search tree that

$$\text{num}(S) = \text{num}_{\text{alt}}(S) \qquad \text{(same)}$$

for all subtrees $S \in \text{Subtrs}(T)$.

Reminder:
Definition. The Search trees $T \in \text{BBTr}$ are defined recursively as follows:

Base case: ($T \in \text{Leaves}$). T is a Search tree.

Constructor case: ($T \in \text{Branching}$). If left(T) and right(T) are both Search trees, and

$$\max(\text{left}(T)) < \text{num}(T) < \min(\text{right}(T)),$$

then T is a Search tree.