This handout recapitulates the binary tree operations we described in class. There is excellent pseudocode for all these operations in most textbooks and online, so I'm only going to write down the ones for which I explicitly gave pseudocode in class.

In all the procedures, \( x \) is a node of the tree. This node holds a key and three pointers: the child pointers \texttt{left} and \texttt{right} and a \texttt{parent} pointer. In this code, we do not distinguish between a tree node and the subtree below it; hence, \( x.\text{min}() \) is the minimum-valued node in the subtree rooted at \( x \).

\texttt{INORDER}(x) prints all the values in the subtree rooted at \( x \) in sorted order from lowest to highest. \texttt{Succ}(x) prints \( x \)'s successor in this sorted order. \texttt{REMOVE}(x) removes a node \( x \) from the tree.
**InOrder**($x$)  
  
  if $x \neq \text{null}$  
  
  InOrder($x$.left)  
  
  print $x$.key  
  
  InOrder($x$.right)  

**Succ**($x$)  
  
  if $x$.$right \neq \text{null}$  
  
  return $x$.$right$.min()  
  
  else  
  
  $y \leftarrow x$.parent  
  
  while $y \neq \text{null}$ and $x = y$.right  
  
  do $\triangleright$ find first leftward edge  
  
  $x \leftarrow y$  
  
  $y \leftarrow y$.parent  
  
  return $y$  

**Remove**($x$)  
  
  if $x$.left = null and $x$.right = null  
  
  remove $x$  
  
  else if $x$.left = null or $x$.right = null  
  
  replace appropriate child of $x$.parent with $x$'s child  
  
  else  
  
  $y \leftarrow \text{succ}(x)$  
  
  $x$.key $\leftarrow y$.key  
  
  $\triangleright$ if other contents besides key, move those too  
  
  Remove($y$)