(defun binary-tree (x)  
"The Data Set, the actual tree"(list (\* 2 x) (+ 1 (\* 2 x))))  
 (defun finite-binary-tree(n)"This Function was given in class.It Limits the tree to the value passed n"  
#'(lambda(x)(remove-if #'(lambda (child) (> child n))(binary-tree x))))  
(defun tree-search (states goal-p successors combiner)"Search algorithm dependant on parameters"  
(print states)  
(cond ((null states) fail)((funcall goal-p (car states)) (car states))(t (tree-search(funcall combiner(funcall successors (car states))(cdr states))goal-p successors combiner))))  
(defun breadth-first-search (start goal-p successors)"Oldest node searched first and expanded nodes are added to the end of thesearch list. Finding node 4 will take 4 turns and finding 66 will take 66"(tree-search (list start) goal-p successors #'swapsuccessors))  
(defun swapsuccessors(x y)"moves the child nodes to the end of the list"(append y x))  
(defun is (value)#'(lambda (x) (eql x value)))