Bst

```
(defstruct node
 (dat nil)
 (left nil)
 (right nil)
)
(defun test()
( princ "empty")
)
(defun new (data)
   (make-node :dat data))
(defun insert(dat h)
 (cond
 ((equal h nil) (new dat))
 ((<(node-dat h) dat)
(setf (node-right h) (insert dat (node-right h)))h)
((>(node-dat h) dat)
(setf (node-left h) (insert dat (node-left h)))h)
))
(defun inorder (tree)
(cond ((equal root nil) (princ "empty")))
(if (null tree ) nil (progn
(inorder(node-left tree))
(format t "~A~%" (node-dat tree))
(inorder (node-right tree))
)
)
)

(defun preorder (tree)
(cond ((equal root nil) (princ "empty")))
(if (null tree )nil
(progn
(format t "~A~%" (node-dat tree))
(preorder(node-left tree))
(preorder (node-right tree))
)))

(defun postorder (tree)
(cond ((equal root nil) (princ "empty")))
(if (null tree)nil
(progn
(postorder(node-left tree))
(postorder (node-right tree))
(format t "~A~%" (node-dat tree))
)))

(defun searc (dat tree)
(cond
((equal tree nil) 0)
((< dat (node-dat tree)) (searc dat (node-left tree)))
((> dat (node-dat tree)) (searc dat (node-right tree)))
((= dat (node-dat tree)) dat)
(t 0)
))
```

```
(defun del (dat tree)
(cond
((equal tree nil) tree)
((< dat (node-dat tree)) (setf (node-left tree) (del dat (node-left tree))) tree)
((> dat (node-dat tree)) (setf (node-right tree) (del dat (node-right tree))) tree)
((and (equal (node-right tree) nil) (equal (node-left tree) nil)) nil)
((equal (node-right tree) nil) (node-left tree))
((equal (node-left tree) nil) (node-right tree))
(t (setf (node-dat tree) (delmin (node-right tree))) (setf (node-right tree) (del (delmin (node-right
tree)) (node-right tree))) tree)
)
)
(defun delmin (tree)
(cond
((equal (node-left tree) nil) (node-dat tree))
(t (delmin (node-left tree)))))
(defun main()
(progn
(setf root nil)
(loop
(format t"~%Menu ~%1.Insert ~%2.Delete ~%3.Preorder ~%4.Inorder ~%5.Postorder ~%6.Search
~%7.Exit ~%")
(princ"Enter the choice:")
(setf a (read))
(cond
((= a 1) (format t "~%Insert ~%Enter the element:") (setf root (insert (read) root)))
((= a 2) (format t "~%Delete ~%Enter the element:") (setf root (del (read) root)))
((= a 3) (format t":~%Preorder ~%") (preorder root))
((= a 4) (format t":~%Inorder ~%") (inorder root))
((= a 5) (format t":~%Postorder ~%") (postorder root))
((= a 6) (format t":~%Search ~%Enter the element") (setf c (read)) (if (=(searc c root) 0) (format t
"~%Not found ~%") (format t "~%found~%")))
((= a 7) (princ "existing......") (return))
(t (format t "~%wrong choice~%"))
))))
(main )
```