

```

#include "tree.h"

TREE *FindInTree (tree, key, compare)
    register TREE *tree;
    char *key;
    int (*compare) ();
{
    register int c;

    while (tree != NULL)
    {
        c = (*compare) (key, KeyOfEntry (tree->entree));
        if (c == 0)
            break;
        else if (c < 0)
            tree = tree->left;
        else
            tree = tree->right;
    }

    return (tree);
}

TREE *InsertInTree (tree, entree, compare)
    register TREE **tree;
    ENTRY *entree;
    int (*compare) ();
{
    register int c;

    while (*tree != NULL)
    {
        c = (*compare) (KeyOfEntry (entree), KeyOfEntry ((*tree)->entree));
        if (c < 0) /* ignore case of repeated keys */
        {
            (*tree)->weight++;
            tree = &((*tree)->left);
        }
        else
            tree = &((*tree)->right);
    }

    if ((*tree = (TREE *) malloc (sizeof (TREE))) == NULL)
        AllocError ("InsertInTree");
    (*tree)->entree = entree;
    (*tree)->left = (*tree)->right = NULL;
    (*tree)->weight = 1;

    return (*tree);
}

TREE *NthInTree (tree, n)
    register TREE *tree;
    register long n;
{
    while (tree != NULL)
    {

```

```
    n -= tree->weight;
    if (n == 0)
        break;
    else if (n < 0)
    {
        n += tree->weight;
        tree = tree->left;
    }
    else
        tree = tree->right;
}

return (tree);
}
```