Burt Rosenberg

Problem Set 3

OUT: 22 SEPTEMBER, 1994
DUE: 6 OCTOBER, 1994

Goals

To learn records and pointers in C. (Note that 2 weeks are given for this assignment.)

Reading Assignment

Read Chapters 5, 6, 8 and 9 from A BOOK ON C.

Programming Assignment

Write a program which converts text to Morse code and Morse code to text. Let us give the program this user interface,

```
morse [-D|-E] [-|filename]
```

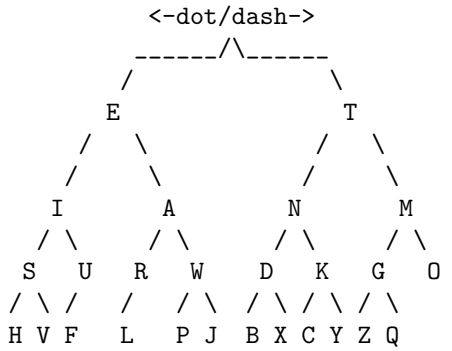
where,

```
-D  switch to tell morse to DECODE
-E  switch to tell morse to ENCODE
-   switch to tell morse to ignore filename and use stdin
filename  file to encode or decode.
output to stdout.
```

furthermore, if the guileless user calls `morse` without any arguments, the program will print a helpful `Usage` line then exit. The input will convert lowercase characters to uppercase, treat uppercase characters as themselves, and consider all other characters as whitespace. The output will transcribe a dot as a period, a dash as an underscore and will separate characters and words with whitespace, either a blank or a newline. Output line length should be controlled to standard limits. Here is the output from *officer pup*,

```
___ ··_· ··_· ·· _·_· · ·_· ·__· ··_ ·__·
```

Encode and decode using this tree representation of the morse code,



Since we are excellent programmers, we will approach the problem of initializing this tree in the proper manner. Write a subroutine taking a *tree description string* and returning a tree described by the string. A tree description string, TDS, has this formal definition,

```
letter ::= @ | A | B | ... | Z
TDS ::= ( TDS letter TDS )
        | ( letter TDS )
        | ( TDS letter )
        | ( letter )
```

This represents a tree according to the formula,

```
 ( T1 l T2 )  =>        l
                      /   \
                     T1   T2  ( subtrees might be omitted )
```

Where l is a character, and T1 and T2 are subtrees which, according to the TDS definition, can either or both be omitted. If omitted the subtree is empty. Then call this subroutine on the TDS for the morse code tree,

```
(((((H)S(V))I((F)U))E(((L)R)A((P)W(J))))@((((B)D(X))N((C)K(Y)))T(((Z)G(Q))M(O))))
```

HINT: Build this tree using left-child, right-child and parent pointers. As usual, an empty child is a Null pointer, as is the parent of the root node. Traverse the tree filling in the array `access_table` with pointers into the tree, `access_table[0]` pointing to the tree node for '@', `access_table[1]` pointing to the tree node for 'A', and so on. For example,

```
struct tree {
    struct tree * left, * right, * parent ;
    int data ; } ;

void init_decode_table( struct tree * t, struct tree * access_table[] )
{
    if (t!=NULL) {
          init_decode_table( t->left, access_table ) ;
          access_table[t->data-'@'] = t ;
          init_decode_table( t->right, access_table ) ;
    }
}
```

Encode by using `access_table` to jump to a starting node in the tree, then emit characters as you climb towards the root. Decode by ingesting characters as you descend the tree starting from the root.

Good luck.