Name: Majed Jamal Majed Hussein

Neptun code: YRGJAF

Morse code

Developer Guide

The presented C program handles Morse code. It decodes from Morse to text and encodes from text to morse.

File formats:

The program handles txt files if wanted. Reading and writing

Structures:

Node:

```
typedef struct node{
    char data;
    char morse; // this will only contain dots and dashs so we could move in the binary tree
    struct node *left, *right;
} node;
```

The building data structure for the binary tree, it contains two char variables, char data will contain any character value, and char morse will contain dots and dashes that will help us with moving in the binary tree, it will also contain two pointers one to the left that will point to nodes that contain dots and one to the right that will point to nodes that contain dashes, and in the case of leaf's the left and the right pointer will point to NULL.

Letter:

```
typedef struct{
    char text;
    char morse[8];
} letter;
```

Contains of a char variable and a string, the char variable will contain any character value, and the string will hold it's equivalent in Morse code.

Functions:

Readuserinput:

```
char *readuserinput()
```

This function allows the user to enter a string of length that he wants, by creating a new dynamically allocated string that has the length of the old string + 1 + 1 ever time the user enters a character than copies the old string into the new one and frees the old string.

crtnode:

```
node* crtnode(char c,char morse){
```

the building function of the binary tree. Contains two parameters, creates a node pointer that has the values passed to it as it's data and morse and returns that pointer.

checkifmorse:

```
int checkifmorse(char *userinput)
```

This function receives a string and checks for every character in that string, if it's not a dot or a dash or a slash then that means that the text is not morse so it will return one else it will return zero.

Encode:

```
void encode(char *input,FILE *fp, letter *pairs)
```

This is the function that turns text into morse code, it first receives input from the user then it receives a file pointer to print to, and then finally it will receive a letter array that contains every character and it's morse equivalent.

It has 3 variables to keep track of the dots and the dashes and the slashes that it will print. First it checks if the input entered is upper case if it is, then it converts it to lower case because morse code does not differentiate between upper case and lower case.

Then it will loop over every char in the letter array, if it finds a match with the user input it will loop over the morse code equivalent of that match and find how many dots and dashes are in it, if the user provides, the file location, then it will print the morse equivalent and the statistics of the morse characters into the file, else it will print them in the terminal.

Decode:

```
void decode(char *input,node *root,FILE *fp)
```

This is the function that turns morse code into text. It receives input from the user, then receives a file pointer to print to before it will receive the root to a binary root.

First, it checks if the input is morse code if not then it returns nothing.

After that, an array is created to keep track of the letters that will be printed by the function.

Then it will create a temp pointer that will move in the binary tree, it will loop over every character in the input, there is a loop inside of it that will end when it encounters a space or the null terminating char or a slash because any of these indicate that it has found a letter, inside the small loop it will check if the input is dot or dash, if dot it moves temp to the left, if dash it moves temp to the right.

after the small loop is completed, it will print the char in the node where the loop stopped in the terminal if the user did not provide a file path, if he did it will print the char to the file, and increase the value of the element that corresponds to the letter in the counter array, and if the user entered a slash than it will print a space in the file or the terminal depending on the user until the big loop is over.

then it will loop over the counter array if it finds a letter that has occurred, it will print it and how many times it has occurred in the terminal or in a file depending on the user.

Insertleft:

```
void insertleft(node* r)
```

This function receives a node pointer than inserts a node that contains a dot in it's morse code to it's left.

Insertright:

```
void inserttight(node* r)
```

This function receives a node pointer than inserts a node that contains a dash in it's morse code to it's right.

deleteTree:

```
void deleteTree(node* root)
```

a recursive function that receives the root of a binary tree transverses through it post-orderly ,deleting every node.

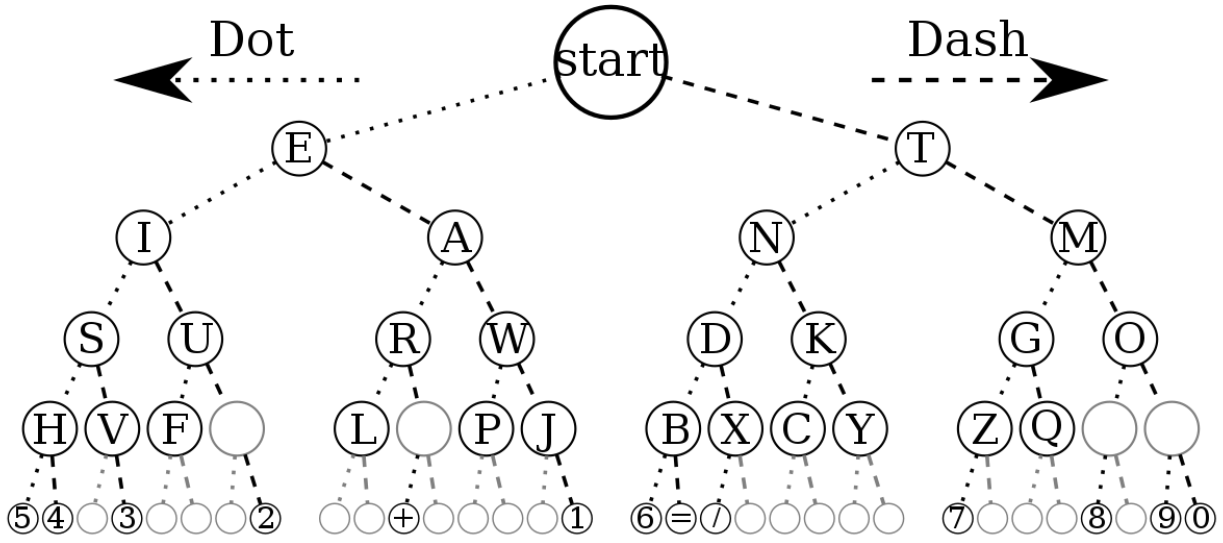
Main:

An array of letters is created that contains every char and it's morse equivalent.

A root for the binary tree is created that has both it's char and it's morse equivalent to NULL,

and create a temp pointer to move in the binary tree,

A binary tree is created per the following diagram,



By looping over every letter expect (space) in the array, inside the loop temp will loop over every dot and dash in the morse part of the letter struct except the last one, in case the morse is a dot then the

temp will check if it has a node on it's left, if it has it will move to it, if not it will create a node containing a dot on it's left then move to it, in case the morse is a dash it will check if it has a node to its right, if It has it will move to if, if not it will create a node that has a dash and then move to it, and it will continue this process for every dot and dash until the small loop is over, than if the last char in morse part is a dot or a dash we move to the left or the right depending on it and check if a node is already there if there is, the value of that node's char data will be assigned to by the same as the letter we are looping over, if there is not then we will create a new node that will have the letter that we are looping over assigned to it char data part.

Then it will prompt the user with choices through a (do while loop).

If user chooses 1:

The user will enter a string through readuserinput() then the string will be encoded and printed in the terminal.

If user chooses 2:

The user will enter a file path and the string in that file will be scanned and encoded than printed in the terminal.

If user chooses 3:

The user will enter a file path and will enter a string through readuserinput(), then the string will be encoded and printed in the file.

If user chooses 4:

The user will enter a string through readuserinput() then the string will be decoded and printed in the terminal.

If user chooses 5:

The user will enter a file path and the string in that file will be scanned and decoded than printed in the terminal.

If user chooses 6:

The user will enter a file path and will enter a string through readuserinput(), then the string will be decoded and printed in the file.

If user chooses 9:

The loop will be broken and then the binary tree will be deleted through deleteTree().