Specification – Morse Code

Purpose

With the use of a binary tree data structure, this project aims to develop a command-line program in C that can encode English text into Morse code, decode Morse code into English text, and yield statistics regarding the Latin and Morse characters encountered in the process. This specification describes the program's expected functionality and behavior.

Usage

The program will be executed from the command line in the following format:

```
morse <switch> <input_file>
```

➔ <switch>: A command-line switch that specifies the operation to be performed. Two valid switches are:

- encode: Encode English text into Morse code.
- decode: Decode Morse code into English text.

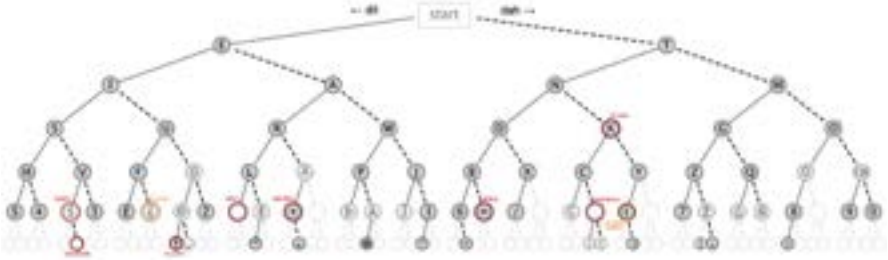
➔ <input_file>: A text file containing the input text or Morse code to be processed. The file must exist, and its path should be provided as an argument. If the file does not exist or if there are any I/O errors, appropriate error messages will be displayed.

Encoding Functionality

- The program should support both uppercase and lowercase English characters.
- The encoding process will convert the English text to Morse code using a predefined mapping that matches each English alphabet character to its Morse code equivalent.
- Example: The character 'A' maps to ".-", 'B' maps to "-...", and so on.

Decoding functionality

- The program will use a dynamically allocated binary tree data structure to decode Morse code back into English text.
- The decoding function will process the Morse code character by character, traversing the binary tree according to the Morse code pattern. It will analyze the code until a space character indicates the end of a Morse code character.
- The decoded text will be stored in an output file and printed to the screen.
- Example: For the Morse code ".- -... -.-.", the output will be "ABC".



Character and Frequency Statistics

The program will generate statistics for both the input and output text.

Statistics will include:

- The total number of Latin characters (letters and spaces) in the input.
- The frequency of each Latin character in the input.
- The total number of Morse code characters (dots, dashes, and spaces) in the output.
- The frequency of each Morse code character in the output.

Error Handling

The program will handle various types of errors, including:

- Invalid command-line arguments: If the switch is not "encode" or "decode," or if the input file is missing, display an error message.
- File I/O errors: If there are issues in reading the input file or writing the output file, display appropriate error messages.

Conclusion

The program encodes and decodes the input text (English text in case of encoding and Morse Code in case of decoding). It offers frequency and character data for the text input and output. A command-line interface with appropriate error handling and effective memory management makes these features available. The file handling is used for storing the input, manipulating it (get the statistics and encode/ decode), and storing the result into an output file (in order to get the statistics of the output).