ECE 36800 – Data Structures
Programming Assignment 4 - Part 2

Purpose of Assignment 4:
Build a Huffman tree for coding and decoding a text file.

Goal of Part 2:
Use a Huffman tree for encoding and decoding a text file.

What to submit:

1. `huffman2.cpp`: Your test program performing the Huffman coding/decoding algorithms. Start with the given file and add your implementation.
2. A word document `proj4p2.docx`: This file should include the printout of your program working with `probability.txt` and `input.txt`. Include a copy (or screenshot) of the printout of the program run.
3. The files `encoded.txt` and `decoded.txt` generated by your program.
4. Push all your files under the "proj4/part2" directory before the deadline.

Other available files:

1. `node.h` and `probability.txt:` same files as in part 1.
2. `input.txt:` The text file that needs to be encoded.
3. `Makefile:` rules to compile the source files for your convenience.

Guideline:
There are four steps in your program (please include your implementation of step 1 and step 2 from part 1):

- Step 1: Read from the file (`probability.txt`) the average occurrence frequencies of 26 English alphabetic letters
- Step 2: Construct a Huffman coding tree and assign Huffman codes for these letters.
  Note: Please have your program print out the code for each letter on screen.
- Step 3: Encode the input file (`input.txt`) using these Huffman codes. Generate an encode file (`encoded.txt`).
- Step 4: Decode the encoded file (`encoded.txt`) to an output file (`decoded.txt`). If your implementation is correct, the output file (`decoded.txt`) should be the same as the input file (`input.txt`).

Please note the following:

– The letters in the table are case insensitive, so when you do encoding, 'A' and 'a' are encoded to the same code.

- It is not required that the non-alphabetic letters are encoded, so letters like ‘ ’ (space), ‘,’, ‘.’, etc. remains the same in the encoded file.
- The decoded file (`decoded.txt`) is case insensitive. You can print the characters all in lower case or upper case.

Example: If ‘A’ is encoded into ‘1’, ‘B’ is encoded into ‘01’, and the input file is “Aa: Bab ab!”, the encoded file would be “11: 01101 101!”. The decoded output file can be “aa: bab ab!” or “AA: BAB AB!”

Suggestions of Implementation:

1. You can use c++ function `bool isalpha(char)` to test if a character is alphabetic. Make sure to include header file `ctype.h` (i.e., `#include <cctype>`).
2. Decoding can be done through the help of the tree (e.g., going left or right if the next code is 0 or 1).

Grading Policy:

Part 2 counts for 50% of the overall points in Programming Assignment 4.

Please make sure your program compiles successfully. If not, 25 points will be deducted.

1. **Executability** (5%):
   - Runtime errors: You program must not have runtime errors (e.g., code crash, infinite loop, reading uninitialized memory, accessing the content of a NULL pointer, etc.).
2. **Programming style** (5%):
   - Code efficiency: Code should use the best approach in every case.
   - Readability: Code should be clean, understandable, and well-organized. Please pay special attention to indentation, use of whitespace, variable naming, and organization.
   - Documentation: Code should be well-commented with file header and comments.
3. **Program Specifications/Correctness** (40%):

   Please refer to the Grading Criteria table for details. Specifically, your program should behave correctly, adhere to the instructions, and pass the test program.

file	**item**	**weight (%)**
huffman2.cpp (and node.h/node.cpp if needed)	correct implementation from part 1	5
	encode input.txt into encoded.txt	10
	decode encoded.txt into decoded.txt	15
proj4p2.docx	running result	10
	total	**40**