2720 Project 3 – Huffman Encoding

Deadline: Friday, April 15th @ 9:00pm
Worth: 100 points

The Assignment

The purpose of this assignment is to introduce you to implementing algorithms that operate on *strings* and compress them. Your grade will be based on the correctness of the program and how well you have documented your program. Please utilize good programming practices and write code in C++ that is efficient and well commented.

Problems

You will implement a Huffman coding system. There are three main parts to this system:

(1) The creation of a Huffman code based upon sample text;
(2) Encoding a message;
(3) Decoding a message.

Your Tasks

For this project, you are required to implement two C++ files named ***encode.cpp*** and ***decode.cpp***. Their executables should be named *encode* and *decode* respectively.

encode.cpp
This is where you should implement parts (1) and (2) of the project (from above). You will be given an input file named *input.txt* as the message that needs to be Huffman encoded. Your program should use the text in input.txt to determine the frequencies of each character in the input file, construct the Huffman Tree based on those frequencies, and create the unique encodings for each character that appears in the text file. Your *encode.cpp* should have TWO output files (see the format section below for details on how they should appear):

- *scheme.txt* – the Huffman encoding scheme
- *message.txt* – the Huffman encoded message

decode.cpp
This is where you will implement part (3) of your project – decoding a message. Here you should use your *scheme.txt* file created in part (1) to decode (decompress) your message in *message.txt* back to a readable message. This should be the same as the original if your code works correctly. The decompressed message should be printed to a file named *ouput.txt.*

NOTE: You must use the exact names of text-files, cpp files, and executables that are specified above - since we will be using an automated grader for this project. Deviations from these provided names may result in a low or failing project grade.

The Format of Sample Input & Output

The input text file (*input.txt*) will contain a single line of characters and whitespace without returns, newlines, or empty lines as shown below:

abc abcde cccc dddd apple

In scheme.txt, you should print the original character, its frequency, and its unique Huffman code. This list should be printed in alphabetical order according to its ASCII value. You are require to keep the following format as:

the single original symbol: *the frequency of the symbol*: *the Huffman code of this symbol*

```
 :4:110
a:3:100
b:2:1011
c:6:01
d:5:00
e:2:1110
l:1:1010
p:2:1111
```

Use ":" to separate the three values; note no additional space, just single ":" here.

In *message.txt*, you should generate a single line of the Huffman encoded message from input.txt. There should be no spaces, newlines, and each bit should be either a 0 or 1.
Ex:
100101101110100101101001110110010101011100000000011010011111111110101110

In *output.txt*, you need to decode *message.txt* back to a readable format using your file, by *scheme.txt*. If your code works correctly, the *output.txt* would equal to *input.txt*.
abc abcde cccc dddd apple

Grading Scheme

This project is worth 100 points and will use an ***automatic grader*** to test your program with multiple file inputs.

You will be graded on the following items:

Correct Length of Huffman-Encoded Message.
Correct Decoding.

Compiling and Submitting

All C++ source code must compile using the g++ compiler located in /usr/bin on nike.cs.uga.edu (nike). You are responsible for checking the correctness of your project before submitting. If you write your program in an IDE, you MUST make sure that it compiles on nike before submitting. If it compiles on your home machine but not on nike, you will receive a zero.

When you are ready to submit, put the source code, its makefile (including compile, run, and clean directives), and a README file in a single directory named **p3** on nike, and use the submit utility to submit the directory to the cs2720 account using the following command: **submit p3 cs2720**. After submitting, you are responsible for verifying that you got the rec (receipt) file. You are also responsible for backing up your code to avoid data loss. Your code may be submitted up to 48 hours late, subject to late penalty deductions per the syllabus.

Only one submission per team. Ensure your README file contains both partners' names, division of labor, and explains what your program does, your design decisions, and how to run the program.