ECE 36800 – Data Structures
Programming Assignment 4 – Bonus

Guideline:

Please implement the following advanced features:

1. Instead of using the input file (`probability.txt`), please calculate the occurrence frequency of all the characters ('a'~'z', 'A'~'Z', and any characters such as ' '(space), ',', '.', etc. as well as special characters like end-of-line) that appeared in the input file (`input.txt`). Generate the Huffman coding tree for all the characters in the input file.
2. Instead of printing character strings of '1' and '0' to the encoded file (`encoded.txt`), please output bit strings only. Therefore, you can observe the size difference between `input.txt` and `encoded.txt`. To implement this feature, you may consider using functions for binary file input/output and bitwise operators (|, &, <<, >>, etc.).
   - An example of binary file I/O:

```
#include <fstream>
#include <sys/stat.h>
...
// Open a file in binary mode for reading
ifsteam readFile(“data1.txt”, ios::in | ios::binary);

// get the size of the input file in bytes
struct stat results;
char* buffer;
if (stat("data1.txt", &results) == 0)
{
     unsigned int size = results.st_size;
     buffer = new char(size);
     readFile.read(buffer, size); // read from a file
}
else // an error occurred
    ...
readFile.close(); // close a file

// Open a file in binary mode for writing
ofstream writeFile;
writeFile.open(“data2.txt”, ios::out | ios::binary);
writeFile.write(buffer, size); // write to a file
writeFile.close(); // close a file
```

   - An example of using bitwise operators.

```
#include <string>
...
string s = “01010111”;
char c = 0x00;
for (int i = 0; i < 8; i++)
{
    if (s[i]==’1’) c = (c<<1) | 0x01; // append 1 in the end
    if (s[i]==’0’) c = (c<<1) | 0x00; // append 0 in the end
}
```

What to submit:

1. `huffman3.cpp`: Your test program performing the Huffman coding/decoding algorithms.
2. A word document `proj4p3.docx`: This file should include the printout of your program working with `input.txt` only. 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/bonus" directory before the deadline.

Grading Policy:

This bonus is worth 20% of Assignment 4.

Note: If your program does not compile, the whole 20 points will be deducted.

1. **Executability** (2%)
2. **Programming style** (2%)
3. **Program Specifications/Correctness** (16%)