****

**BLG 372E**

**ANALYSIS OF ALGORITHMS II**

CRN: 22853

**REPORT OF HOMEWORK #2**

Submission Date: 26.04.2014

**STUDENT NAME: TUĞRUL YATAĞAN**

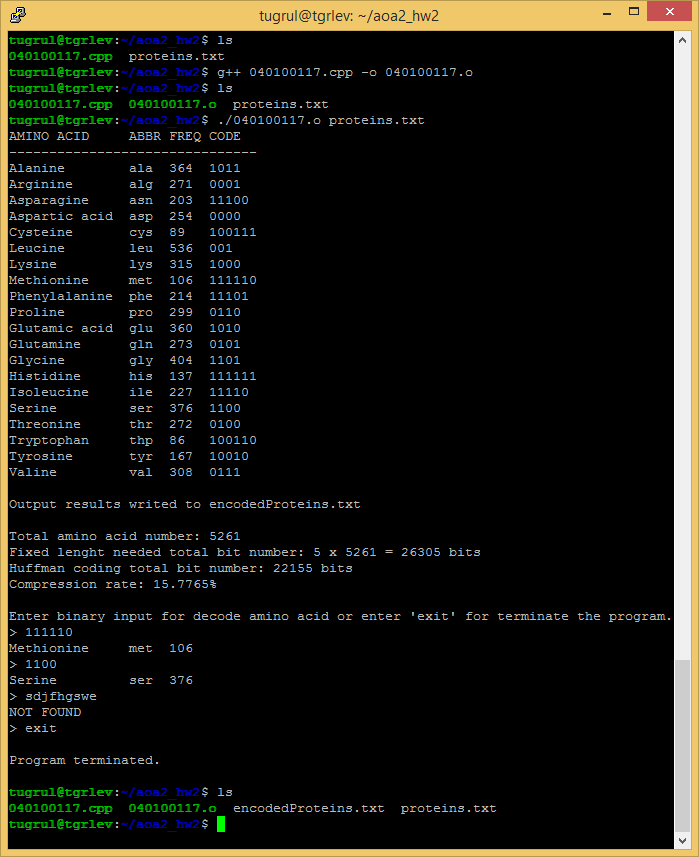
**STUDENT NUMBER: 040100117**

# **Building and Running**

The program built and compiled without any warning or error under g++ and the program executed with commands:

g++ 040100117.cpp –o Huffman

./Huffman proteins.txt

Sample output is below:

# **Data Structures and Variables**

Purpose of the classes and methods explained in the source code as comment lines.

* AminoAcid aminoAcids[20];

array of amino acids which keeps; name, abbreviated name, occurrence number and Huffman code of amino acids.

* HuffmanTree Tree;

Keeps tree data structure of Huffman tree and linked list.

* HuffmanNode;

Node class for keeping right, left child pointer for tree and next pointer for linked list.

* HuffmanNode \*listHead;

Head of the linked list to be converted to Huffman tree. Linked list generated from aminoAcids array and it uses for tree building operation.

* HuffmanNode \*treeRoot;

Root of the Huffman tree.

* LookupTable Table;

Lookup table class which keeps trie data structure for lookup operation.

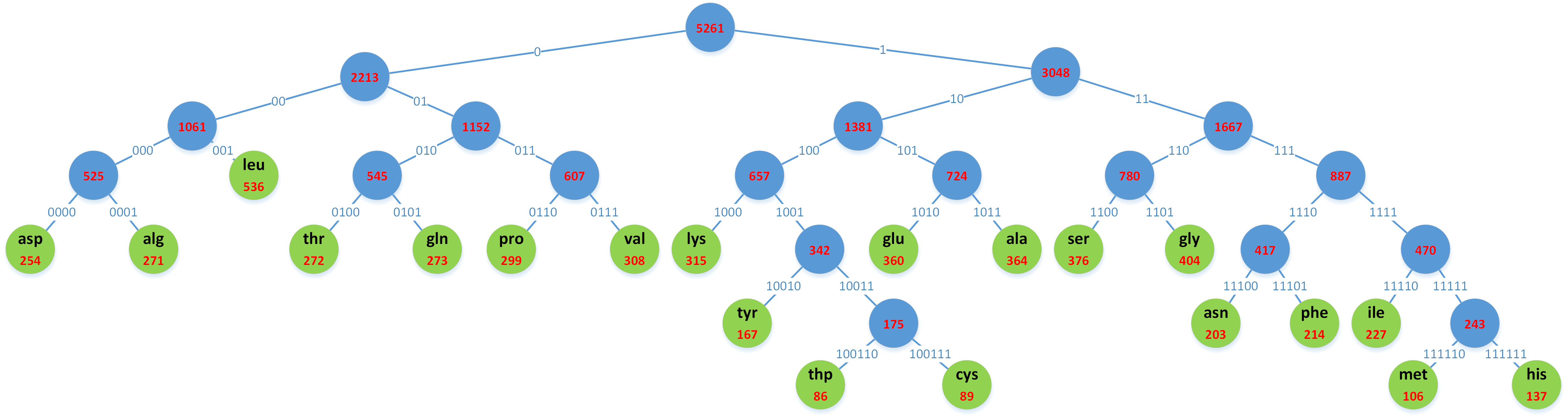
* LookupNode;

Node of lookup table trie data structure which keeps letter and children information.

* LookupNode \*lookupRoot;

Root of the trie data structure of lookup table

Final Huffman tree is below:



# **Analysis**

Total amino acid number in the input file is 5261. If we use fixed length codes for these 20 amino acid items, we need to denote each one of them with 5 bits (24<20<25). As a result, we need 5261 x 5 = 26305 bits to represent the input. However, Huffman coding uses 22155 bits. Compression rate is:

If the input file consists of elements which has much higher frequency over other elements and elements which has smaller frequency, the compression rate will be higher. So if distribution of occurrence number of elements are scattered then the compression rate will be higher, if distribution of occurrence number of elements are close to between them (exp: all elements has same occurrence number) then the compression rate will lower.