1. ***Designing a flexible tree structure.***

**(a, b, c)**

* *Design a tree structure that you will implement and subclass to solve both the Huffman Coding Problem and the Splay Tree problem.* *Include a class diagram showing all the supporting classes of this tree structure. Explain your decisions.*
* The goal was to design both the Tree and Node super classes as abstract and extend them into their appropriate Huffman and Splay implementations (Huffman, SplayTree, and HuffmanNode, SplayNode).
* This tree super class contains a size attribute and a pre- and post-order traversal method common to all trees.
* Both extended trees compose their node counterparts as a protected root attribute.

**(d, e)**

* *Include a class diagram showing how you Huffman Coding implementation will extend your general tree structure. Explain your decisions textually.*
* Used a priority queue (PriorityQueue) instead of using a simple array and re-sorting on each add/remove/find operation since it is more efficient (log*n* in the worst case) as opposed to sorting an entire array (*n*, *n2*, or *n*log*n*) in the worst case.

**(f, g)**

* *Include a class diagram showing how your Splay Tree implementation will extend your general tree structure. Explain your decisions textually.*
* See last page of this document for the UML diagram.
* **(h)** *Explain the advantages of using a Splay Tree over an AVL Tree to solve question 3, referencing your diagrams and decisions as is appropriate.*
* See 4. (b).

1. ***Textual responses.***

**(a)** *Huffman.*

* *Encode the string associated with your student id and record the result here.*

she advised him to come back at once.

1110010001111001001101010101001110111110011111010100100011011010110010110110001010000110010111111000111110011010000101011001001101100011011010101000111111001011

* *Describe what percentage fewer bits were used than the fixed-length ASCII encoding would have taken.*
* It would take log2(34) or 6 bits to provide the 34 unique codes needed to represent the 34 characters. This amounts to 6×37 or 222 total bits for fixed-length encoding. In contrast, using Jabberwock.txt text to create the Huffman encoding, the total number of bits required for the string given above would require only 160 total bits. This is a space savings of ~28%, i.e. Huffman encoding requires ~72% of the total number of bits compared to fixed-length encoding.
* *Indicate whether you feel that the encoded string matched the source text frequencies, and provide your reasoning.*
* Since there is a space reduction (less required total bits) to encode the string, this implies that the string does match (to some extent) the source text; otherwise, there would have been hardly any reduction to using fixed-length encoding, if any at all.

**(b)** *Advanced trees.*

* *Provide the output you recorded based on the Operations file provided in question 3.*

49471 compares

2862 Zig-Zigs

2621 Zig-Zags

* *Explain whether you feel that the Splay Tree was the correct choice, given that you have now implemented and tested it against the provided representative data. Explain anything you have learned through implementing and testing that would affect any future designs (for example, might you use an AVL tree).*
* As long as the same BST is being acted upon over a series of many operations (search, add, delete) large enough (which in our case, the series of operations is), the guaranteed average cost of *k*log*n* using a Splay implementation, where *k* is the number of operations, its use is justified over say a AVL based implementation. Although AVL implementations have the advantage of a guaranteed cost of log*n* for each operation, unless very few operations are performed on the same tree, a Splay implementation has an equivalent, if not better, performance in the average multi-operation case.

