Ferdinand Mudjialim, Partner: Nick Parisi
Data Structures - Riley

Project 3 Pseudocode: Huffman and Djikstra's Pseudocode

```
Djikstra(vertices, adj_list, weights, starting vertex):
    S;  # Dictionary of vertex:weight aka shortest path distance
    Q;  # Priority Queue
    For vertex in vertices:
        Add vertex into S with weight infinity
    Set starting vertex in S with weight 0
    Add (0, starting vertex) into Q
    While Q is not empty:
        Dequeue a vertex from Q and set u = vertex
        For v in adj_list of u:
        If current weight of v in S > current weight of u in S + distance from u to v:
            Set current weight of v in S = current weight of u in S + distance from u to
v
            Add (weight of v in S, v) into Q
    Return S
```

-=-

```
Huffman(chars, freqs):
    S;  # Dictionary of characters to codewords
    Q;  # Priority Queue
    For char in chars:
        Set codeword of char in S to '-' initially
    # Constructing the Huffman Tree
    For letter in freqs:
        # Note: Tree has (key, value) and right / left children
        Add (frequency of letter from freqs, Tree(frequency … freqs, letter)) to Q
    While size of Q is not 1:
        N1 = Q.dequeue()
        N2 = Q.dequeue()
        T = Tree( N1[0] + N2[0], '')  # New 'blank' tree with frequency sum but no letter
        T.left = N1[1]  # first letter
        T.right = N2[1]  # second letter
        Add (N1[0] + N2[0], T) into Q  # Frequency sum and new constructed tree
    # Traversing the Huffman Tree for encodings
    huffmanTree = Q.dequeue()[1]
    # traverse function traverses tree, adding 0 for left side and 1 for right side into the
    # encoding dictionary S if the node has a letter (not just frequency sum)
    traverse(huffmanTree)
```

Return S