VGP310 Assignment #2
W. Dobson 1-24-2018

Write a program where the user enters a string that is scanned for character frequencies stored in a *MinHeap Priority Queue* that is used to generate a *Huffman Decoder Tree* which can then be used to create a related *Huffman Encoder Table*.

Once the encoder table and decoder tree are created, use these to:
1. Encode the original input string into an encoded bitstream
2. Decode the encoded bitsteam back to the original text string.

Be sure to display the contents of the *encoder table*, *encoded bitstream*, and *decoded string* along with the lengths of the *original string*, *encoded bitstream*, and *decoded string*. For convenience we will use a string of '0' and '1' characters for the encoded bitstream in this project noting that they can be easily converted to binary bits and "packed" into bytes for storage or transmission in a real world application..

You project will use a code symbol node structure (CsymNode)

```
struct CsymNode{
     char c;        // data character
     unsigned f;    // freq of that characters
     CsymNode *left, *right;
}
```

You should implement functions:

// Function to scan a string counting frequency of characters then putting them
// into a priority queue. Returns the number of codesymbols discovered.

```
unsigned FreqCount(string& str, priority_queue,CsymNode*,
vector<CsymNode*>, compare_Csysm>& pq)
```

// Function to build a Huffman coding tree

```
CsymNode *huffmantree(priority_queue<CsymNode*, vector<CsymNode*>,
compare_Csym>& pq)
```

// Recursive function to traverse Huffman tree and build an encoder table (dictionary)

```
void huffencodetable(CsymNode *root, vector<EncoderNode> *etable,
string str)
```

// Huffman encoder function creates a bitstream string of 0 and 1 char representing
// the binary encoded data (using a string out of convenience).

```
void huffmanencoder(string& str, vector<EncoderNode>& etable, string&
bitstream)
```

// Huffman decoder function takes a bitstream string input and decodes
// using decoder tree referenced by root ptr.

```
void huffmandecoder(CsymNode *root, string& bitstream, string&
decodestr)
```

The `main()` code body should instantiate the data structures/objects needed then call the functions in sequence to perform the following tasks.