COLBY COLLEGE
Computer Science Department

CS231 **Allen B. Downey**
Fall 1999

Assignment 11: Huffman Codes

Due: Wednesday 1 December

Reading: Standish Chapter 8.11

For this assigment you are going to write a program that reads a sample string and constructs a Huffman code for it. The Huffman code will be encapsulated in an object; one of the operations you can perform on this object is `encode`, which takes a String and encodes it.

In order to get the whole thing working, you need three pieces:

- A frequency table (basically a histogram of characters).
- A Huffman tree (as seen on pages 299-301).
- A code table, which contains the Huffman code for each character.

Build the Frequency table

1. Go to `http://rocky.colby.edu/cs231/code/hw11/Huffman.java/` and pick up a copy of the stripped version of the Huffman and FreqTab classes.
2. Look over the code: I have provided the outline of the class definitions, including interfaces for the methods. There are comments indicating which methods you should fill in.
3. Test your program on a couple of sample strings.

Build the Huffman tree

1. Create a new class named `HuffTree` to represent a Huffman Tree (as seen on pages 299-301). In this case we won't bother with a sentinel (just to keep the number of classes reasonable).

   ```
   public class HuffTree implements Comparable {
     int freq;
     char c;
     HuffTree left, right;
   }
   ```

   Since HuffTrees implement the Comparable interface, they can be inserted into a Heap. You can use your Heap implementation or mine, but you should not have to make any changes to Heap.java or Comparable.java—that's the nice thing about ADTs!

2. Add the the methods `buildHuffTree` and `decode` to the Huffman class.

   To debug your program you may want to use methods that print the contents of the tree. Keep in mind that you might not get exactly the same tree as in the book, but it should be similar in the sense that each letter should have the same code length as in the book, even if the code is not identical.

Build the Code Table

1. Create a new class named `CodeTab` that contains an array of codes. The codes will be represented as Strings containing 1's and 0's.

2. Add the methods `buildCodeTab` and `encode` to the Huffman class. `encode` should take a String and return a String of 1's and 0's that represents the String according to the appropriate Huffman code.