In this project, we will use trees to construct a variable-length encoding of the symbols of a finite alphabet Σ. Each symbol will be represented by a series of bits. The basic idea is to use fewer bits for symbols which occur frequently, and more bits for symbols which occur infrequently. This enables us to encode strings of symbols using fewer bits than would be used by a fixed-length encoding. Our method depends on knowing the probability distribution of the occurrence of the various symbols in the alphabet. For example, among words in the Concise Oxford Dictionary, approximately 12.7% of the letters are “e”, while 0.153% of the letters are “j”.¹

We illustrate the method by example. Consider the following table of symbols and corresponding probabilities:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability</td>
<td>0.32</td>
<td>0.20</td>
<td>0.18</td>
<td>0.13</td>
<td>0.12</td>
<td>0.05</td>
</tr>
</tbody>
</table>

We start with a “forest” of trees; each tree consists of a single node labeled with the given probabilities:

```
A  B  C  D  E  F
0.32 0.20 0.18 0.13 0.12 0.05
```

**Step 1:** On each step, we select the two trees with the smallest probability labels. In this step the two smallest are 0.12 and 0.05 (corresponding to “E” and “F”). We create a new node and link those two as sub-trees. We label the new node with the sum of the probabilities in the sub-trees. We now have:

```
A  B  C  D
0.32 0.20 0.18 0.13
```

Notice that at the end of step 1, we have five trees instead of six.

**Step 2:** The two smallest probabilities are 0.13 and 0.17. Once again, we create a new node and link these two as sub-trees. After labeling the new node, we now have:

```
A  B  C  D  E  F
0.32 0.20 0.18 0.13 0.12 0.05
```

¹Source: https://en.wikipedia.org/wiki/Letter_frequency
**Step 3:** The two smallest probabilities are now 0.20 and 0.18.

```
A   B   C   D   E   F
0.32 0.20 0.18 0.13 0.12 0.05
```

**Step 4:** The two smallest probabilities are now 0.32 and 0.30.

```
A   B   C   D   E   F
0.32 0.20 0.18 0.13 0.12 0.05
```

**Step 5:** On the final step, only two trees remain. We merge them into one tree as shown below.

```
A   B   C   D   E   F
0.32 0.20 0.18 0.13 0.12 0.05
```

We use the paths from the root node to the leaves to define a sequence of bits which represent each letter. We construct our code by labeling each left link with a 0, and each right link with a 1. A labeled diagram is shown on the following page.

A code constructed this way has a very important property: **No code is a prefix of any other code.** This makes decoding a bit stream into individual letters very simple. We read bits until it matches one of our codes: no looking back and second guessing.
We read the codes from the tree diagram above by the labels found on the links as we go from root to leaf. I.e.:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>00</td>
</tr>
<tr>
<td>C</td>
<td>01</td>
</tr>
<tr>
<td>D</td>
<td>110</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
</tr>
</tbody>
</table>

**Expected Value of Number of Bits Per Symbol**

We compute the expected value of the number of bits $x$ to represent one symbol as follows:

$$E(x) = 0.32 \cdot 2 + 0.20 \cdot 2 + 0.18 \cdot 2 + 0.13 \cdot 3 + 0.12 \cdot 4 + 0.05 \cdot 4 = 2.47$$

A fixed-length code would need at least 3 bits to represent the six possible letters in our alphabet. Our code achieves a compression ratio of $3/2.47 \approx 1.2146$.

**Your Task:** Implement the encoding scheme illustrated above. The input is a file containing a table of symbols and their corresponding probabilities. For the example illustrated above, the input would be:

```
A 0.32
B 0.20
C 0.18
D 0.13
E 0.13
F 0.05
```

Your output must be a table of symbols and a corresponding code. Skip a blank line and output the expected value of the number of bits required to represent one symbol. For example:
A 10
B 00
C 01
D 110
E 1110
F 1111

2.47

Turn-In:
Keep all your work in a sub-directory named Lab3. Change to the parent directory of Lab3 (use the command cd ..) and create a tar archive of your work using the command:

```bash
% tar cf Lab3.tar Lab3
```

Upload the file Lab3.tar to your account on telesto.