2006 Canadian Computing Competition
Day 1, Question 3

Input file: `codec.in` (for `compress`), `compress.out` (for `decompress`)
Output file: `compress.out` (for `compress`), `codec.out` (for `decompress`)
Source files: `n:\codec\compress.{c, cpp, pas}`
`n:\codec\decompress.{c, cpp, pas}`

Codec

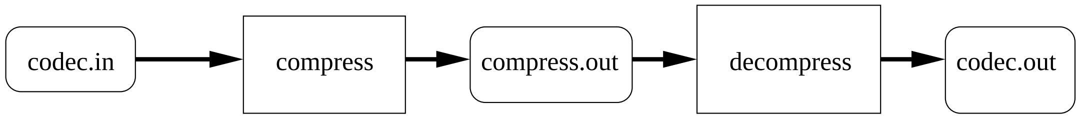
The problem of lossless data compression is to transform some data into a compressed form such that:

(a) the original can be reproduced exactly from the compressed form.

(b) the compressed form is as small can reasonably be achieved.

You are to write two programs – `compress` that performs lossless compression and `decompress` that reproduces the original data from the compressed form. The data to be compressed will be plain English text represented using printable ASCII characters (i.e., all characters with ASCII values between 32 and 126 inclusive). The compressed form is a string of binary bits. For convenience, we will represent this string of bits as a character string containing only 0s and 1s.

`compress` reads the original data from the file `codec.in` and writes the compressed form `compress.out`. `decompress` reads the compressed data from a file called `compress.out` and writes the corresponding original data to `codec.out`. Of course, `codec.in` and `codec.out` must be the same file. Pictorially, we have the following flow of information:



`compress` must output only 0s and 1s and `decompress` must exactly reverse the effects of `compress`. That is, condition (a) above must hold for any English text. If `compress` and `decompress` meet these criteria, your score will be determined by the relative size of the input and output by

$$\text{score (as \%)} = 50 \cdot \sqrt{\frac{8c - b}{c}},$$

where c is the total number of characters in the original text and b is the number of bits in the compressed form. Note that scores may exceed 100%, but scores that are less than 0 will be given 0% (i.e., no negative marks will be given, but bonus marks may be awarded).

Discussion and Hints

It is well known that any ASCII character can be represented using 8 bits. Such a representation would achieve a score of 0 using the formula above. Since there are fewer than 128 possible symbols in the input, it is possible to represent each one with 7 bits. Such a representation would receive a score of at least 50%.

A smaller representation can be achieved, with high probability, by observing that some letters are more common than others. Suppose we estimate that a character α occurs with probability p_α in a given context. The best possible code will use $-\log_2(p_\alpha)$ bits to represent that character. If one estimates p_α one can construct a *prefix* code with about $-\log_2(p_\alpha)$ bits for each character in the following manner:

- build a binary tree with one leaf for each character α
- organize the tree so that the depth of α is approximately $-\log_2(p_\alpha)$
- use a binary representation of the path (0=left, 1=right) to represent α in the compressed data.

One way to estimate p_α is simply to compute the fraction of characters equal to α in a sample of data similar to that to be compressed. Another is to use an adaptive method, in which the data is compressed one character at a time, and the sample consists of the text already compressed. A sample of English text is available in the file `sampleText.txt`.

It is also possible to estimate p_α using the context in which it occurs; for example, in English a "q" is very likely to be followed by a "u" (e.g., quick, quack, quit, quiz, but not qiviut, which happens to be the wool of a musk-ox).

Use this information, or any other information at your disposal, to build the best Compressor and Decompressor you are able.

Input

The input to `compress` will consist of n characters ($1 \leq n \leq 1000000$), as described above.

The input to `decompress` will consist of m 0s and 1s ($1 \leq m \leq 8n$).

Output

The output of `compress` will be a sequence of 0s and 1s, with no other characters (i.e., no newline characters should be outputted).

The output of `decompress` is a sequence of at most 1000000 uppercase letters, lowercase letters, spaces, newlines and punctuation symbols.

Sample Input (to `compress`)

```
To be or not to be?
```

Possible Output for Sample Input (`compress`)

```
0111001000010110000100110100001100010011110000011110010000101101111 1
```

Sample Input (to `decompress`)

```
01110010000101100001001101000011000100111100000111100100001011011111
```

Output for Sample Input (`decompress`)

```
To be or not to be?
```

Explanation

The sample compressor systematically uses the following codes for each of the input characters:

`<space>`	`000`
`o`	`010`
`n`	`01100`
`r`	`01101`
`T`	`01110`
`t`	`011110`
`?`	`011111`
`b`	`10`
`e`	`11`

The compressed output uses these codes, as shown below:

```
01110010000101100001001101000011000100111100000111100100001011011111
TTTTTooo   bbee   ooorrrrr   nnnnnoootttttt   ttttttooo   bbee??????
```