Assignment 2
(100 points)

RULES GOVERNING ACADEMIC INTEGRITY ARE STRICTLY OBSERVED WITHOUT EXCEPTION.

Purpose:

To allow you to exercise with a logical thinking process to formulate very simple algorithms, leading the way to the implementation of this logic in some programming language like Python. The logic will possibly include simple inputs and outputs, lists, strings, control flow.

Question:

You are to implement what is called a dictionary-based compression algorithm that works as follows:

The following example illustrates how the encoding is performed on an input file that is composed of only the letters (a) and (b). Assume there are no spaces in the sentence below, but the spaces are inserted to make the text more readable.

Given the input string a b b a a b b a a b a b b

Step 1: The dictionary will be initially composed of all the letters in the file:

Index	Entry
0	a
1	b

Start traversing your input string:

a b b a a b b a a b a b b

Step 2: Find the longest block of words W that appeared in the dictionary above. This longest block is simply letter “a” because it appears in index 0 of the dictionary. A will thus be encoded using its index 0.

Step 3: encode W (happens to be “a”) with its index (0) in the dictionary.
a b b a a b b a a b a b b
0

Step 4:
Add W followed by the first symbol of the next block to the dictionary. The next symbol is “b”. W, which is “a” followed by “b” is now “ab”. Add “ab” to the dictionary.

The dictionary now is as follows:

Index	Entry
0	a
1	b
2	ab

Repeat step (2) again.

The next longest string already in the dictionary is the string "b".

B will be encoded using its index, 1

a b b a a b b a a b a b b

0 1

Add "b" along with the next letter "b" to the dictionary. The dictionary is now:

Index	Entry
0	a
1	b
2	ab
3	bb

The next longest string in the dictionary now is "b" again!. It will be coded using its index, 1.

a b b a a b b a a b a b b

0 1 1

Add "b" along with the next letter to the dictionary. The next letter is "a". So the dictionary becomes:

Index	Entry
0	a
1	b
2	ab
3	bb
4	ba

The next longest string in the dictionary is now "a". It will be encoded using the index of "a" in the dictionary, which is "0".

a b b a a b b a a b a b b

0 1 1 0

Add "a" along with the next letter, "a" to the dictionary, it becomes:

Index	Entry
0	a
1	b
2	ab
3	bb
4	ba
5	aa

The next longest string in the dictionary is now “ab”, it will be encoded now using the index of “ab” in the dictionary, which is 2.

a b b a ab b a a b a b b
0 1 1 0 2

And so on …

The General Encoding Algorithm

1. Initialize the dictionary to contain all blocks of length one.
2. Search for the longest block **W** which has already appeared in the dictionary.
3. Encode **W** by its index (location) in the dictionary.
4. Add **W** followed by first the symbol of the next block to the dictionary.
5. If not done, go to (2).

A practical size of the dictionary is 4096 entries. This compression algorithm works well with longer files. Try it with a text file, it should work. You are to compute the compression ratio: The size of the input file / the size needed for compression (including the size of your dictionary!). Print the compression ratio.

Your input is provided as a text file composed of any characters. You are to ask for the name of the input file, and an output file. You will also ask the user whether they want to encode or decode the file. When you encode and then decode, the output must be identical to the original file.

You will be graded as follows:

- 5 points for getting input and output file names, as well as encoding/decoding.
- 10 points for computing and printing the compression ratio.
- 10 points for building dictionary properly.
- 20 points for proper encoding.
- 20 points for proper decoding (text file similar to original file)
- 10 points for accommodating all characters.
- 5 points for documentation.
- 10 points for using functions.
- 10 points for a professional evaluation of your code.