SCHOOL OF SOFTWARE ENGINEERING OF USTC

FIRST SEMESTER, 2012–2013
Campus: Suzhou

AGILE SOFTWARE DEVELOPMENT

Error management in *C*
Build with Waf

(Time allowed: TWO hours)

The objective of this experiment class is to practice some Python and taste what coding with a scripting language feels like. To do so, we will implement a fairly simple compression algorithm : *byte pair encoding*. This algorithm gives fair performance on text data. You would be wise to check at least the Wikipedia entry for *byte pair encoding*.

The work have to be done in Python. The version of Python does not matter, 2.x or 3.x are accepted. You have to work alone, one student per computer, this is non-negociable. It is to help you to be focused and practice by yourself. If you are stuck after trying various ways, you can call for help. You might have to patient while the teacher is busy helping someone else.

1. In one line of code, turn a string into a tuple of ASCII codes. The function that gives the ASCII code of a character is *ord()*

2. Write a function *get_unused_code_list(data)* that returns the list of the unused ASCII codes in the *data* string. A reminder : ASCII codes goes from 0 to 255. This function can be written in a single line of code.

3. Write a function *get_pairs(data)* that **generates** the pairs of characters from the sequence *data*. For instance, with the string *abcde*, the pairs *(a, b)*, *(b, c)*, *(c, d)*, *(d, e)* would be generated. This function can be written in a single line of code. You would be wise to make this function works with any kind of sequence, not just strings.

4. Write a function *count_pairs(data)* that returns a dictionary where

- the keys are pairs from the sequence *data*
- the values are how many times the key appears in the input sequence *data*

For instance, with the string *abababc*, the returned dictionary would be *ab : 3, ba : 2, bc : 1*. You would be wise to make this function works with any kind of sequence, not just strings.

5. Write a function *get_most_frequent_pair(data)* that returns the most frequent pair in a string *data*. Using the previously written function, it can be done in a single line of code. You would be wise to make this function works with any kind of sequence, not just strings.

6. Write a function *replace_pair_by_code(data, replace_pair, replace_value*, that **generate** a copy of the sequence *data*. In this copy, any occurence of the pair *replace_pair* is replaced by *replace_value*. For example *replace_pair_by_code('abababc', 'ab', 'X')* would generate *['X', 'X', 'X', 'c']*

CONTINUED

7. You made it to the seconde page of this assignement ! Great ! You can now implement the function *compress(data)*, that compress the input string *data* by replacing its most frequent character pair, by one ASCII code not present in the string. The function returns the compressed string.

8. You can improve *compress(data)* by simply repeating the same operation (replacing the most frequent pair of a string by an unused ASCII code), until no compression is possible. Return the compressed string, and the list of replacements the function found.

9. Write an function *compress(compressed_data, replacement_list)*, that undo the work done by a call to *compress(data)*. *uncompress* should return exactly the imput of *compress*.