Tokenization – Lab 2 Handout

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded. The tokens become the input for another process like parsing and text mining.

Tokenization is used in computer science, where it plays a large part in the process of lexical analysis.

Tokenization relies mostly on simple heuristics in order to separate tokens by following a few steps:

- Tokens or words are separated by whitespace, punctuation marks or line breaks
- White space or punctuation marks may or may not be included depending on the need
- All characters within contiguous strings are part of the token. Tokens can be made up of all alpha characters, alphanumeric characters or numeric characters only.

Tokens themselves can also be separators. For example, in most programming languages, identifiers can be placed together with arithmetic operators without white spaces. Although it seems that this would appear as a single word or token, the grammar of the language actually considers the mathematical operator (a token) as a separator, so even when multiple tokens are bunched up together, they can still be separated via the mathematical operator.

Lab Task:

1. Here, in this lab, you will be carrying out tokenization on strings. Write a piece of code in C/C++ that takes an input string from user on command line and then asks for the

tokenizer on which the string is to be tokenized. The program then tokenizes the string based on the given tokenizer. The input tokenizer can be a white space or any alphanumeric character. The output should be in the form of tokens, each on a new line, as shown below.

Input	**Tokenizer**	**Output**
She sells sea shells at the seashore.	s	he_ ell _ ea_ hell _at_the_ ea hore.
10 Humpty Dumpty sat on a wall.	(white space)	10 Humpty Dumpty sat on a wall.
I, along with him, on a seashore, in the summer, bought coconut juice.	, (coma)	I _along_with_him _on_a_seashore

_ (underscore) represents white space here.

		_in_the_summer _bought_coconut_juice.

NOTE: Your code should not use any built-in libraries/functions for tokenization.

2. After Tokenization, you will be arranging the tokens into ascending order based on their first character. For example, special characters such as punctuation marks, white spaces are displayed first, then the alphanumeric characters are displayed, digits first, followed by the alphabetical display or characters.

Tokenized String	**Arranged Tokens**
he_ ell _ ea_ hell _at_the_ ea hore.	_ _at_all_ ell ea_ ea he_ hell hore.
10 Humpty Dumpty sat on a	10 a Dumpty Humpty on sat

wall.	wall.
I _along_with_him _on_a_seashore _in_the_summer _bought_coconut_juice.	_along_with_him _on_a_seashore _in_the_summer _bought_coconut_juice. I

Note: Arranging the tokens is not case sensitive.

_ (underscore) represents white space here.