Problem 1 – Lempel Ziv Algorithm

In this problem, we will implement the sliding-window Lempel–Ziv (often abbreviated as SWLZ or LZ77) compression algorithm.

(a) Write a program for a function `MatchLengthPosition(window, text)`, whose inputs are:

- a "window" of string, denoted by the variable `win`, and
- a text sample string, denoted by the variable `text`.

Let us index the symbols in the string `win` to start from '0' at the rightmost symbol, and increase to | `win` |-1, which is the leftmost symbol. We say that a "match" has occurred, if for some $0 \leq i \leq$ | `win` |-1, the substring $(win(\mathrm{i}), win(0))$matches with some substring of `text`, starting from the leftmost symbol of `text`. If such a match occurs, the function returns the following outputs:

- a flag bit 1,
- the starting position,i, of the **longest** match present in the window, and
- the match length.

If there is no match, this function returns a flag bit 0, and the first character of `text`.
For example,
`[1,2,3]=MatchLengthPosition(''MY '',''MY MY WHAT A HAT IS THAT'')` and
`[0,'B']=MatchLengthPosition(''AAAA'',''BABBAA'')`.

(b) Using the function in part (a), write a program for a function `ParseSWLZ(InputText, WindowSize)` that takes the following inputs:

- a string of characters, denoted by the variable `InputText`, and
- a positive integer, denoted by the variable `WindowSize`

The program then returns the encoding by the sliding window Lempel–Ziv algorithm (using a window of size `WindowSize`).
For example, if `InputText=''MY MY WHAT A HAT IS THAT''` and `WindowSize=16`, then the function should output the following:

(0,M), (0,Y), (0,-), (1,2,3), (0,W), (0,H), (0,A), (0,T), (1,4,1), (1,2,1), (1,1,1), (1,5,4), (0,I), (0,S), (1,2,1), (1,4,1), (1,7,3),

where the '-' character denotes a space.