Exercise 2: Tokenizer

Gabinete, Keith Ginoel S.
CD-1L

Instructions:

Create a program (using any Programming Language) that:

- reads a file
- tokenize its content (separate the words, symbols, etc, ...)
- print all the tokens
- print the number of tokens

Provide the following

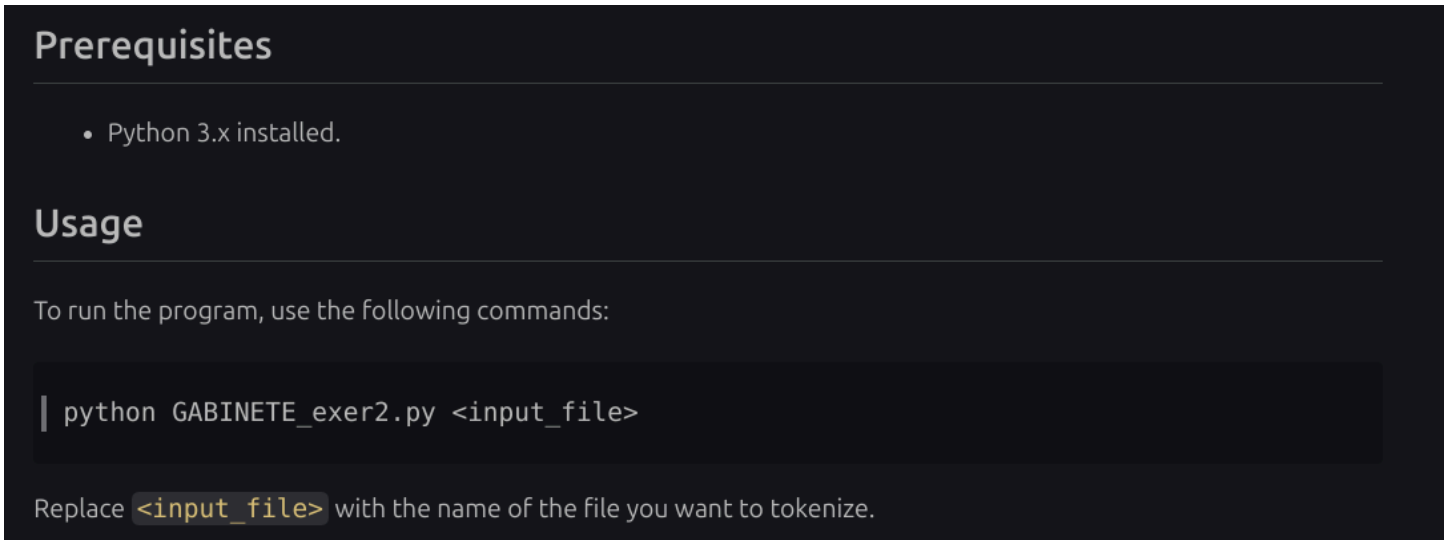
- source code
- Instructions on how to compile and/or run the program
- screenshot

You can check your program by tokenizing the sample files attached. The answer should be the following:

sample1.txt has 7 tokens
sample2.txt has 27 tokens

1. Source Code: View **GABINETE_exer2.py**

2. Instructions on how to run the program (Screenshot from **README.MD**):



3. Screenshots

```
shio@X509FB:~/Desktop/repos/principles-of-compiler-design/02 - Tokenizer$ python3 GABINETEKG_exer2.py sample1.txt
INPUT:
-------------------------------------------------------
int max(int i);
-------------------------------------------------------
TOKEN               TAG
-------------------------------------------------------
int                 Integer Type
max                 Identifier
(                   Opening Parenthesis
int                 Integer Type
i                   Identifier
)                   Closing Parenthesis
;                   Statement Terminator

TOTAL:              7
shio@X509FB:~/Desktop/repos/principles-of-compiler-design/02 - Tokenizer$
```

```
shio@X509FB:~/Desktop/repos/principles-of-compiler-design/02 - Tokenizer$ python3 GABINETEKG_exer2.py sample2.txt
INPUT:
-------------------------------------------------------
int main()
{
  int a = 10, b = 20;
  printf("sum is :%d",a+b);
  return 0;
}
-------------------------------------------------------
TOKEN               TAG
-------------------------------------------------------
int                 Integer Type
main                Main Function
(                   Opening Parenthesis
)                   Closing Parenthesis
{                   Opening Curly Brace
int                 Integer Type
a                   Identifier
=                   Assignment Operator
10                  Integer Literal
,                   Comma Separator
b                   Identifier
=                   Assignment Operator
20                  Integer Literal
;                   Statement Terminator
printf              Print Function
(                   Opening Parenthesis
"sum is :%d"        String Literal
,                   Comma Separator
a                   Identifier
+                   Addition Operator
b                   Identifier
)                   Closing Parenthesis
;                   Statement Terminator
return              Return Statement
0                   Integer Literal
;                   Statement Terminator
}                   Closing Curly Brace

TOTAL:              27
shio@X509FB:~/Desktop/repos/principles-of-compiler-design/02 - Tokenizer$
```