Chappuit louis-maximilien charles jean

1820253065

Gkikas-panousis georges

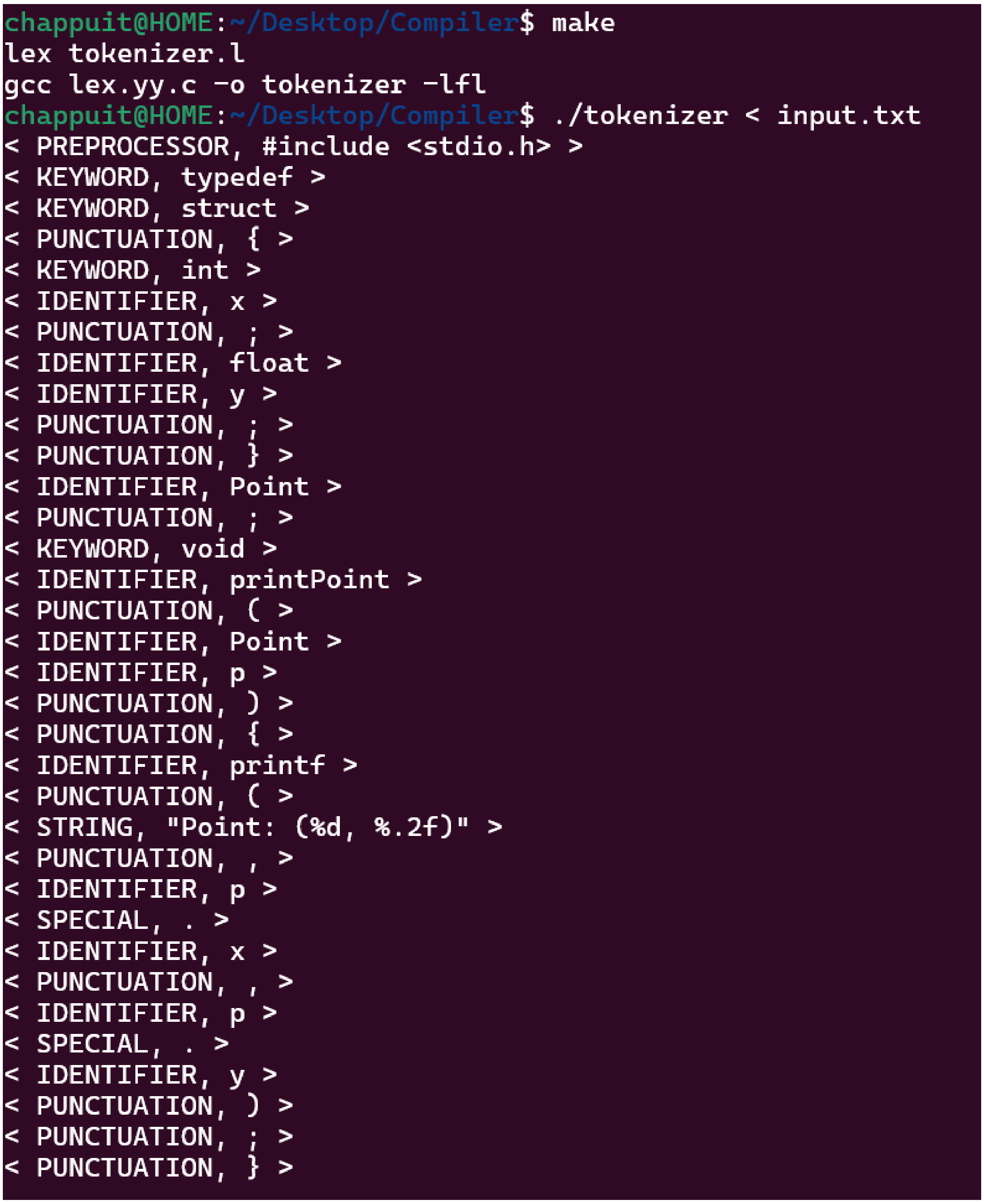
1820253079

Dyoilis anastasios

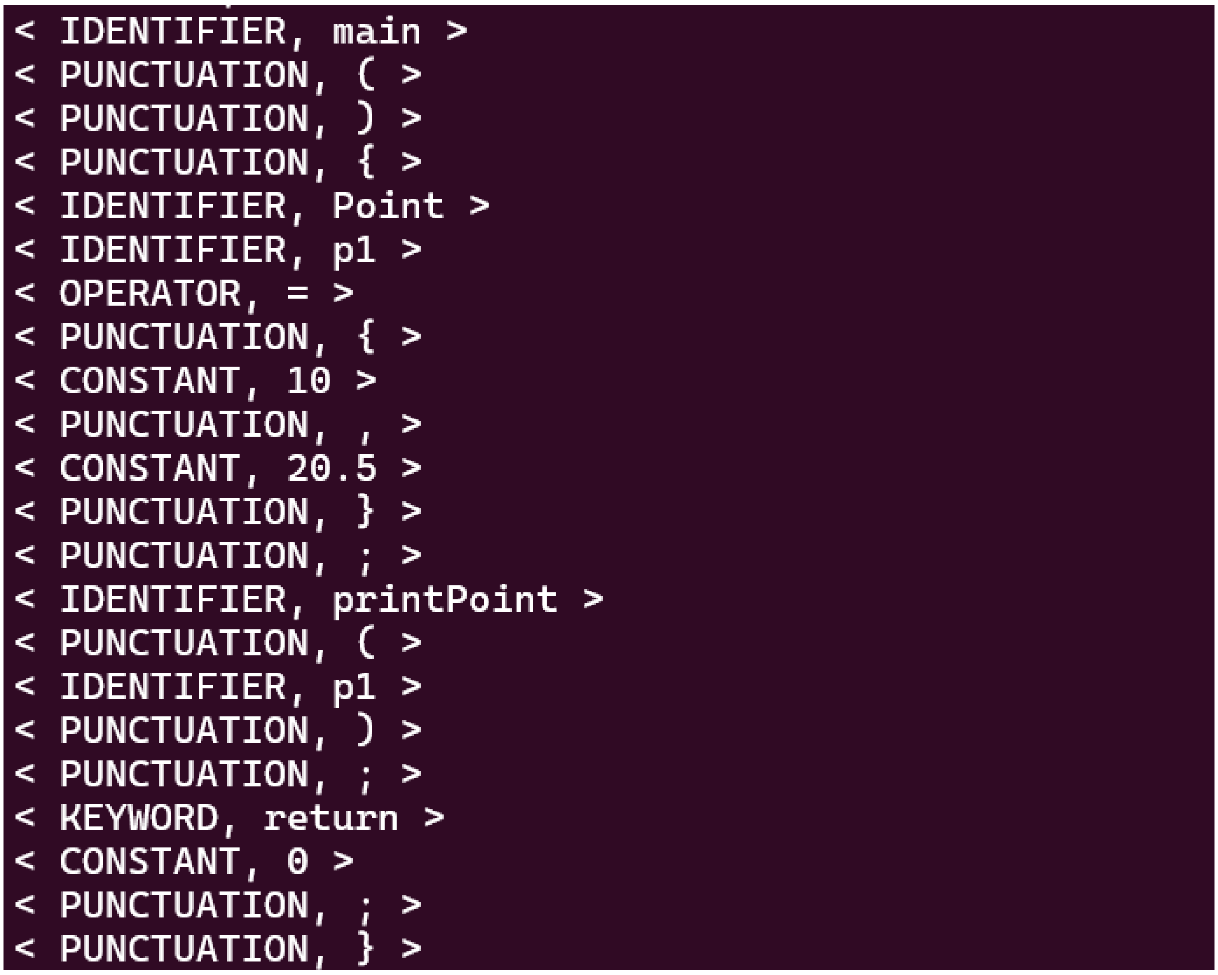
1820253075

REPORT FOR THE LAB 3

Here is the screenshots with the example given by the lab3 document.

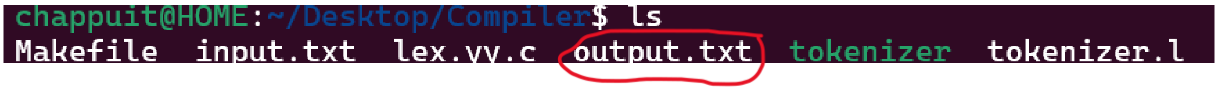


```
chappuit@HOME:~/Desktop/Compiler$ make
lex tokenizer.l
gcc lex.yy.c -o tokenizer -lfl
chappuit@HOME:~/Desktop/Compiler$ ./tokenizer < input.txt
< PREPROCESSOR, #include <stdio.h> >
< KEYWORD, typedef >
< KEYWORD, struct >
< PUNCTUATION, { >
< KEYWORD, int >
< IDENTIFIER, x >
< PUNCTUATION, ; >
< IDENTIFIER, float >
< IDENTIFIER, y >
< PUNCTUATION, ; >
< PUNCTUATION, } >
< IDENTIFIER, Point >
< PUNCTUATION, ; >
< KEYWORD, void >
< IDENTIFIER, printPoint >
< PUNCTUATION, ( >
< IDENTIFIER, Point >
< IDENTIFIER, p >
< PUNCTUATION, ) >
< PUNCTUATION, { >
< IDENTIFIER, printf >
< PUNCTUATION, ( >
< STRING, "Point: (%d, %.2f)" >
< PUNCTUATION, , >
< IDENTIFIER, p >
< SPECIAL, . >
< IDENTIFIER, x >
< PUNCTUATION, , >
< IDENTIFIER, p >
< SPECIAL, . >
< IDENTIFIER, y >
< PUNCTUATION, ) >
< PUNCTUATION, ; >
< PUNCTUATION, } >
```



```
< IDENTIFIER, main >
< PUNCTUATION, ( >
< PUNCTUATION, ) >
< PUNCTUATION, { >
< IDENTIFIER, Point >
< IDENTIFIER, p1 >
< OPERATOR, = >
< PUNCTUATION, { >
< CONSTANT, 10 >
< PUNCTUATION, , >
< CONSTANT, 20.5 >
< PUNCTUATION, } >
< PUNCTUATION, ; >
< IDENTIFIER, printPoint >
< PUNCTUATION, ( >
< IDENTIFIER, p1 >
< PUNCTUATION, ) >
< PUNCTUATION, ; >
< KEYWORD, return >
< CONSTANT, 0 >
< PUNCTUATION, ; >
< PUNCTUATION, } >
```

What is printed is also written in the output.txt file:



```
chappuit@HOME:~/Desktop/Compiler$ ls
Makefile  input.txt  lex.yy.c  output.txt  tokenizer  tokenizer.l
```

If the ouput.txt file can't be opened,then an error is returned.

Let's talk about regular expression that are more complex than "[0-9]":

0[xX][0-9a-fA-F]+ for Hexadecimal numbers like "0xFA32D677".

\'.\' accept any element from the alphabet between ' and '.

\"(\\.|[^\"])*\" \\. Include echaped characters (\n for example), [^\"] for all characters excepted the " for obvious reasons. And we repeat that as much as we need.

"/*"([^*]|[\r\n])*"*/" is for multiple line commentaries. [^*] all characters excepted the star,[\r\n]for new line or "retour chariot". and the " | " in the middle will chose either [^*] or [\r\n] as much time as we need (using of *).

"#".* simply the preprocessor regular expression that takes all came after a #.