

## TP Flex Exercices

1- Ecrire un analyseur lexical qui compte le nombre de caractères, le nombre de mots et le nombre de ligne dans un texte en utilisant yyleng.

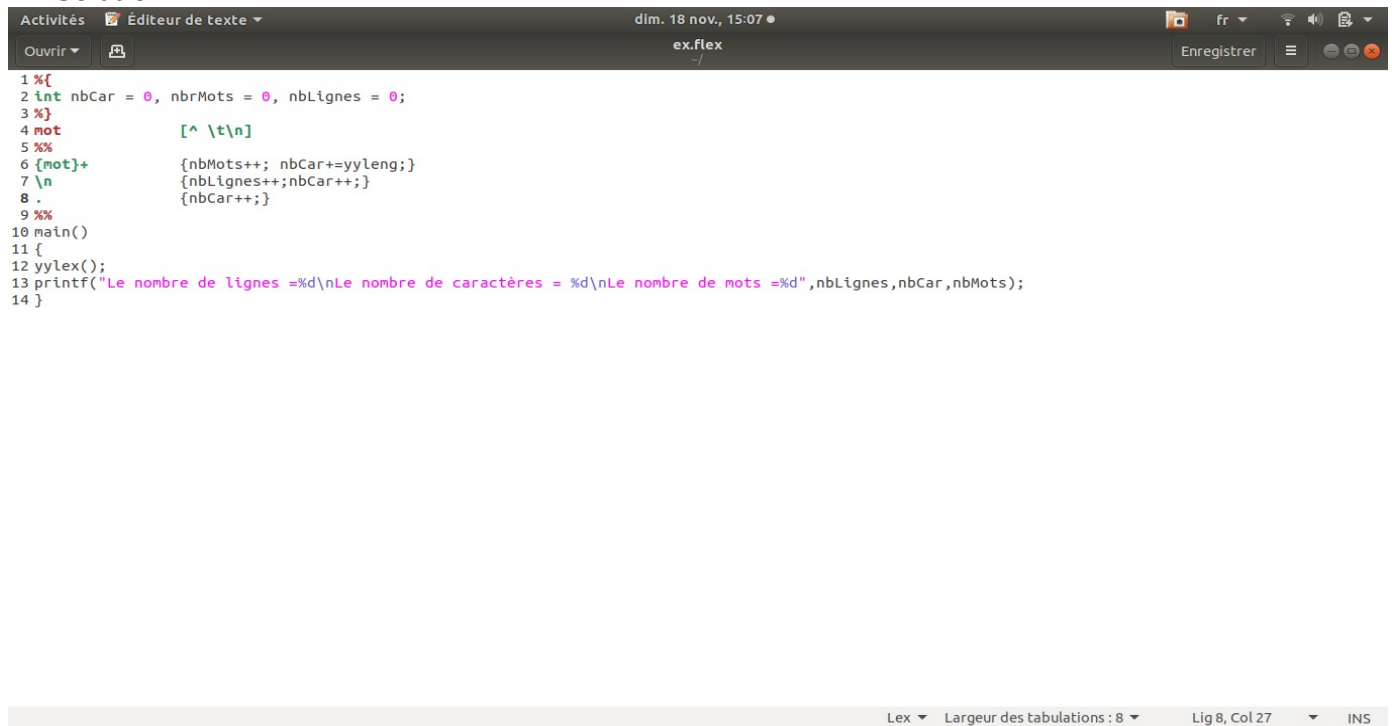
### Solution 1



```
1 %{
2 int nbCar = 0, nbrMots = 0, nbLignes = 0;
3 %}
4 chiffre      [0-9]
5 lettre       [a-zA-Z]
6 blanc        " "
7 alphanum     ({chiffre}|{lettre})
8 %%
9 {alphanum}+  {nbCar+=yyleng; nbrMots++;}
10 {blanc}+     {nbCar+=yyleng;}
11 \n          {nbLignes++;}
12 %%
13 main()
14 {
15 yylex();
16 printf("Le nombre de lignes =%d\nLe nombre de caractères = %d\nLe nombre de mots =%d",nbLignes,nbCar,nbrMots);
17 }
```

Lex ▾ Largeur des tabulations : 8 ▾ Lig 15, Col 9 ▾ INS

### Solution 2



```
1 %{
2 int nbCar = 0, nbrMots = 0, nbLignes = 0;
3 %}
4 mot          [^ \t\n]
5 %%
6 {mot}+       {nbrMots++; nbCar+=yyleng;}
7 \n          {nbLignes++;nbCar++;}
8 .           {nbCar++;}
9 %%
10 main()
11 {
12 yylex();
13 printf("Le nombre de lignes =%d\nLe nombre de caractères = %d\nLe nombre de mots =%d",nbLignes,nbCar,nbrMots);
14 }
```

Lex ▾ Largeur des tabulations : 8 ▾ Lig 8, Col 27 ▾ INS

## 2- Ecrire un analyseur lexical qui fait la somme des entiers saisies au clavier



```
1 %{
2 #include<stdlib.h>
3 int somme = 0;
4 %}
5
6 %%
7 [0-9]+      {somme + = atoi(yytext);}
8 .          {}
9 %%
10 main()
11 {
12 yylex();
13 printf("La somme des entiers saisies = %d \n",somme);
14 }
```

Lex ▾ Largeur des tabulations : 8 ▾ Lig 14, Col 2 ▾ INS


## 3- Ecrire un analyseur lexical qui compte le nombre de mots commençant par la lettre V (majuscule).



```
1 %{
2
3 int nb = 0;
4 %}
5 mots      [a-z]+
6 %%
7 V{mots}   {nb++;}
8 {mots}|({mots}"V"{mots}) {}
9 [^A-Z]    {}
10
11 %%
12 main()
13 {
14 yylex();
15 printf("Le nombre de mots qui commencent avec un V = %d \n",nb);
16 }
```

Lex ▾ Largeur des tabulations : 8 ▾ Lig 9, Col 6 ▾ INS

4- Ecrire un analyseur lexical qui insert un numéro de ligne pour chaque lignes d'un fichier.

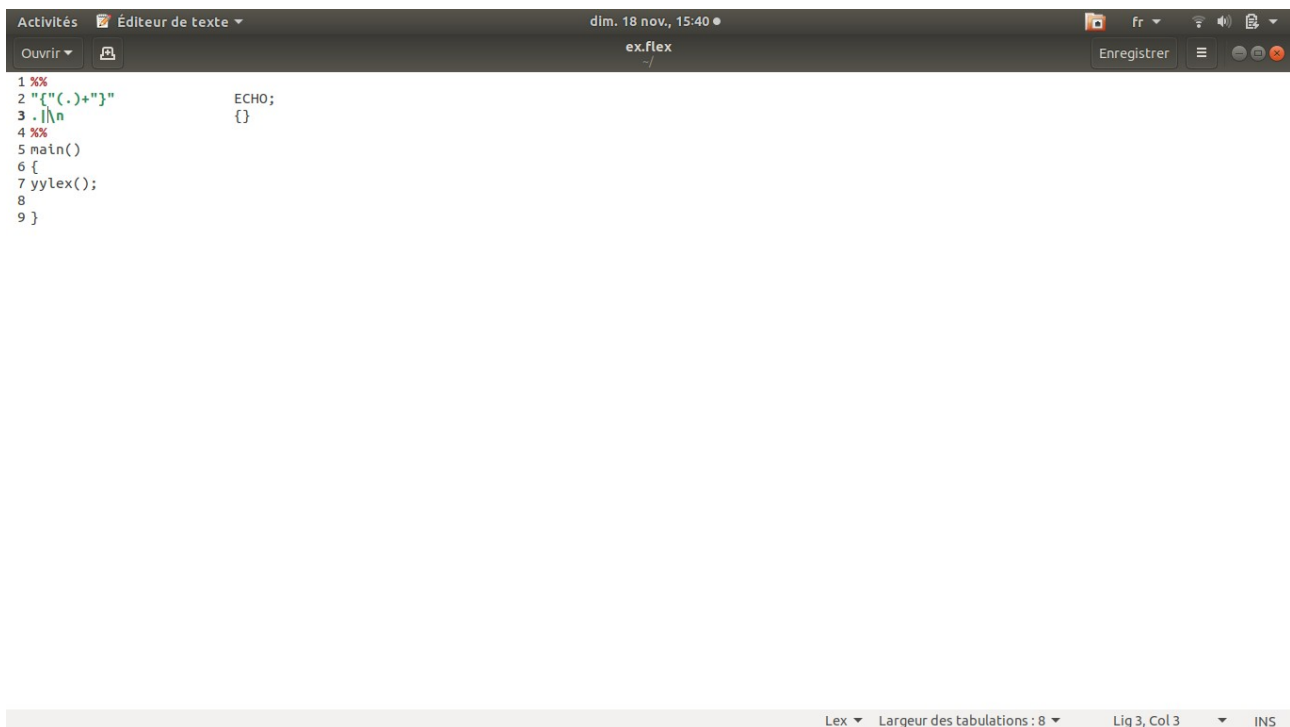


The screenshot shows a text editor window titled "ex.flex" with the following C code:

```
1 %%  
2 ^(.*)\n      {printf("%4d\t%s",yylineno++,yytext);}  
3  
4 %%  
5 int main(int argc, char **argv)  
6 {  
7     yyin = fopen(argv[1], "r");  
8     yylex();  
9     fclose(yyin);  
10 }
```

The status bar at the bottom indicates "Lex", "Largeur des tabulations : 8", "Lig 7, Col 20", and "INS".

5- Ecrire un analyseur lexical qui n'imprime que le texte compris entre { et }

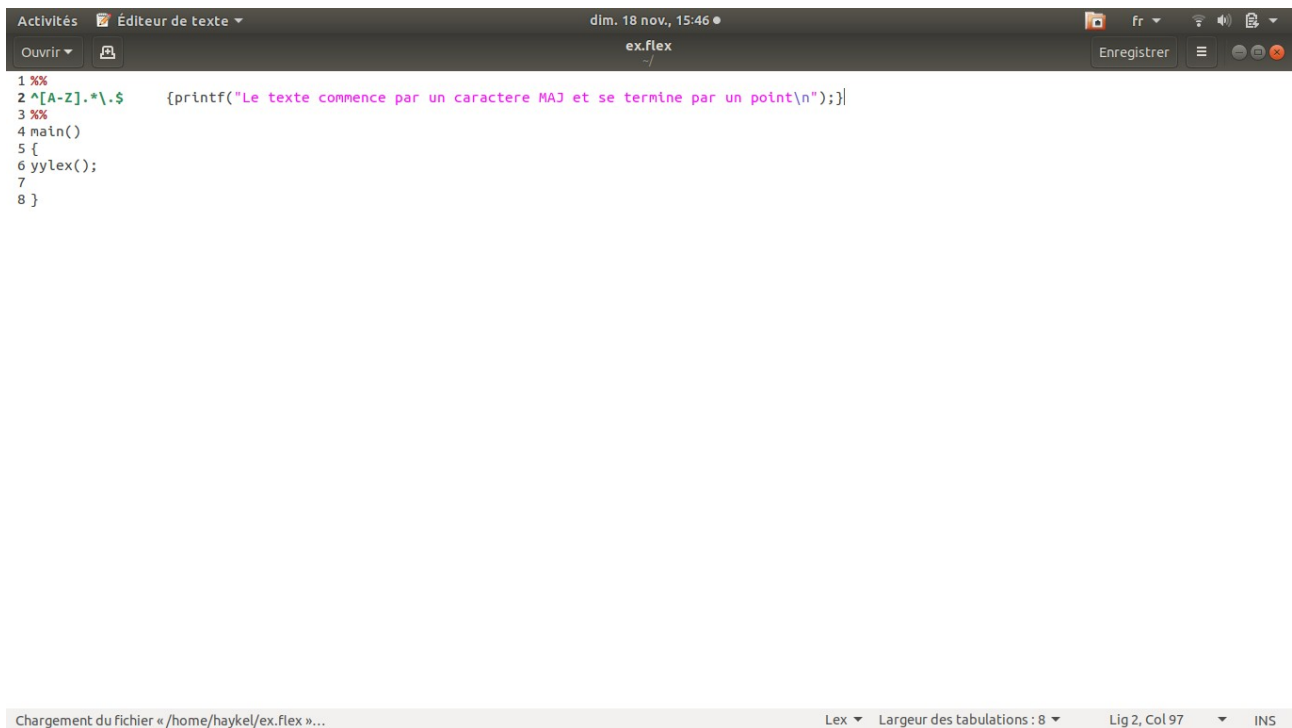


The screenshot shows a text editor window titled "ex.flex" with the following C code:

```
1 %%  
2 "{(.)+}"      ECHO;  
3 .|\n      {}  
4 %%  
5 main()  
6 {  
7     yylex();  
8  
9 }
```

The status bar at the bottom indicates "Lex", "Largeur des tabulations : 8", "Lig 3, Col 3", and "INS".

6- Ecrire un analyseur lexical qui vérifie si un texte commence par un caractères majuscule et se termine par un point.

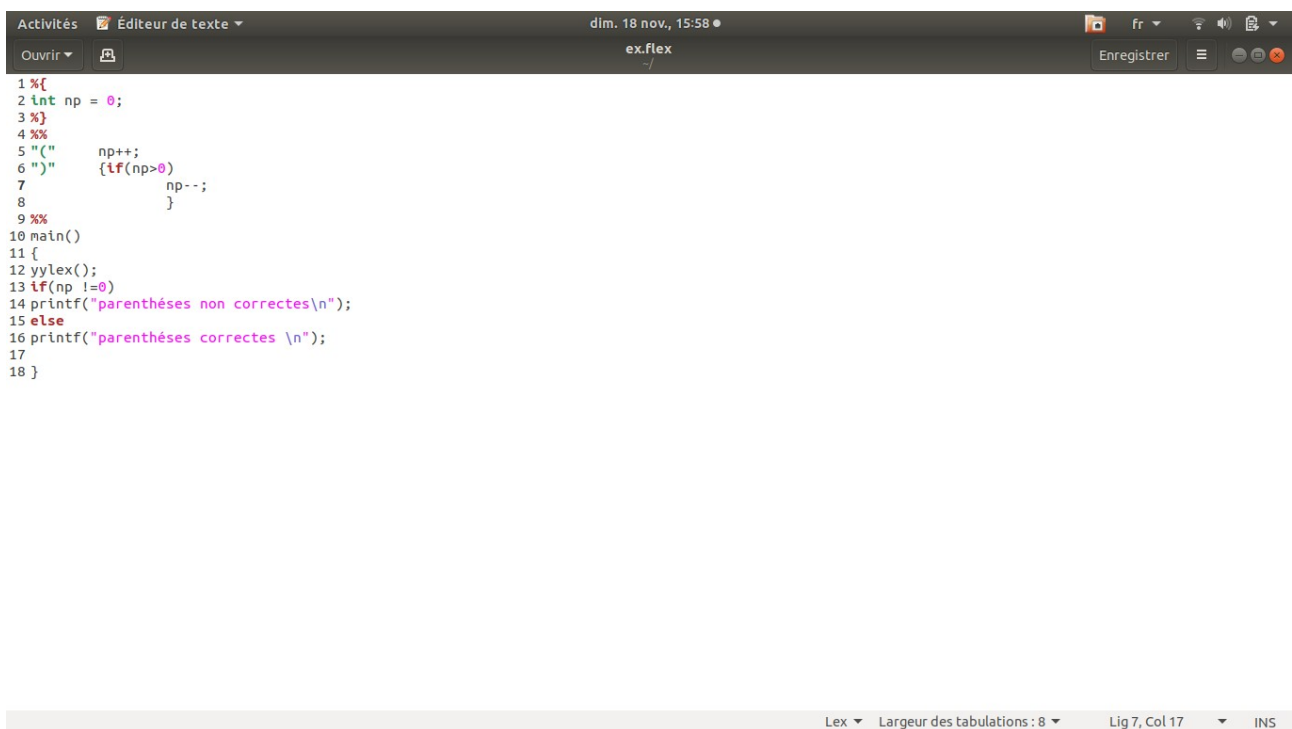


```
1 %%
2 ^[A-Z].*\.$ {printf("Le texte commence par un caractere MAJ et se termine par un point\n");}
3 %%
4 main()
5 {
6 yylex();
7 }
8 }
```

Chargement du fichier « /home/haykel/ex.flex »...

Lex ▾ Largeur des tabulations : 8 ▾ Lig 2, Col 97 ▾ INS

7- Ecrire un analyseur lexical qui vérifie le bon parenthésage d'un flot de données.



```
1 %{
2 int np = 0;
3 %}
4 %%
5 "(" np++;
6 ")" {if(np>0) np--;}
7
8 %%
9 main()
10 {
11 {
12 yylex();
13 if(np !=0)
14 printf("parenthèses non correctes\n");
15 else
16 printf("parenthèses correctes \n");
17 }
18 }
```

Lex ▾ Largeur des tabulations : 8 ▾ Lig 7, Col 17 ▾ INS