The lexical analyzer generator — Lex

Exercise 1.

- Write and compile the following specification file:

```
%{
int nb;
%}
pairpair (aa|bb)*((ab|ba)(aa|bb)*(ab|ba)(aa|bb)*)*
%%
{pairpair}    { printf("[%s] : even number of a′s and b′s \n",yytext);}
a*b*          { printf("[%s] : a′s first and then b′s\n",yytext);}
[\n]          // nothing is done
.             { nb++; printf("(ignore %c)\n", yytext[0]); }
%%
main()
{
      nb =0;
      yylex();
      printf("\n%d ignored characters \n",nb);
}
```

- test the previous Lex program on the inputs; babbaaab ,abbb, aabb, baabbbb ,bbaabbba, baabbbbab, aaabbbba . . .

  Try to guess what the program will output before it does!

- Same question with the two lines swapped:

```
...
a*b*           {printf("[%s]:a′s first and then b′s \n",yytext);}
{pairpair}     {printf("[%s]:even number of a′s and b′s \n",yytext);}
...
```

  Is there a difference? Which one? Why?

Exercise 2.

Write a Lex program that takes as input a file of integers and outputs the integers with **3** added to all numbers divisible by **7**. Integers not divisible by **7** should remain unchanged.

Hint:

Use the **atoi()** function, which converts a string of digits into its numeric value

Exercise 3.

Write a Lex program that counts the number of vowels, consonants, and punctuation characters in a text entered from the keyboard.

Exercise 4.

Write a Lex program that:

a) Converts text written in uppercase letters to lowercase.

b) Removes spaces and tabs at the end of each line.

Hint:
Use the **`tolower()`** function, which converts an uppercase character to lowercase.

.