Name: Ahmed Ali Abd El Meguid Ali Hassanine ID: 7

Name: Hazem Morsy Hassan Morsy ID: 18

Name: Sherif Mohamed Mostafa Mohamed ID: 22

Name: Youssef Mohamed Fathy Hassan Aly ID: 66

"Phase1"

"Bonus task"

Problem statement:

The required is to use a tool generating a lexical analyzer for given regular expressions, provide steps for using it and also screen shots.

Used tool:

Flex (free implementation of Lex program).

Steps using LEX:

- The input is a file (.l extension) containing 3 sections separated by the delimiter '%%'
    - **The first** section (definition section) has some (optional) information as Flex table size (and other constants), variable declarations, manifest constants and regular definitions etc. enclosed by % { definitions } % and anything in this brackets is processed by Flex.
    - **The second** section represents the rules (regular expressions). It is enclosed by %% pattern action %% where patterns can be like a* and action is a c code enclosed by {} on the same line.
    - **The third** section (also optional) is user C code (some user defined program).
- Then, using the CMD we can run the command flex (filename.l) to generate a lex.yy.c file corresponding to the scanner (lexical analyzer defined by the written code in the input file).

- We then run this file using gcc lex.yy.c and a a.exe file is generated which is an executable fille that can take input patterns (code) and generate the corresponding actions (tokens defined).
- The input to the exe can be given through the CMD or read from a file.
- Here I use a file for convenience.
- Note: we have to add the bin folder location from the installed package to the paths defined to the system to be able to run flex command

Example patterns used in Flex and their meaning:

Pattern	Meaning
[0-9]	all digit between 0 and 9
[0+9]	either 0, + or 9
[0, 9]	either 0, ‘, ‘ or 9
[0 9]	either 0, ‘ ‘ or 9
[-09]	either -, 0 or 9
[-0-9]	either – or all digit between 0 and 9
[0-9]+	one or more digit between 0 and 9
[^a]	all the other characters except a
[^A-Z]	all the other characters except the upper case letters
a{2, 4}	either aa, aaa or aaaa
a{2, }	two or more occurrences of a
a{4}	exactly 4 a’s i.e, aaaa
.	any character except newline
a*	0 or more occurrences of a
a+	1 or more occurrences of a
[a-z]	all lower case letters
[a-zA-Z]	any alphabetic letter
w(x | y)z	wxz or wyz
dog	Match string “dog”
a.*b	Any string starts with a and ends with b
[^a-z]+	Any string of one or more char but doesn’t have a lower case letter

Symbols used:

- ^: but, +: one or more, *:0 or more, ?: optional, x-y: from x to y, |:or

First run:

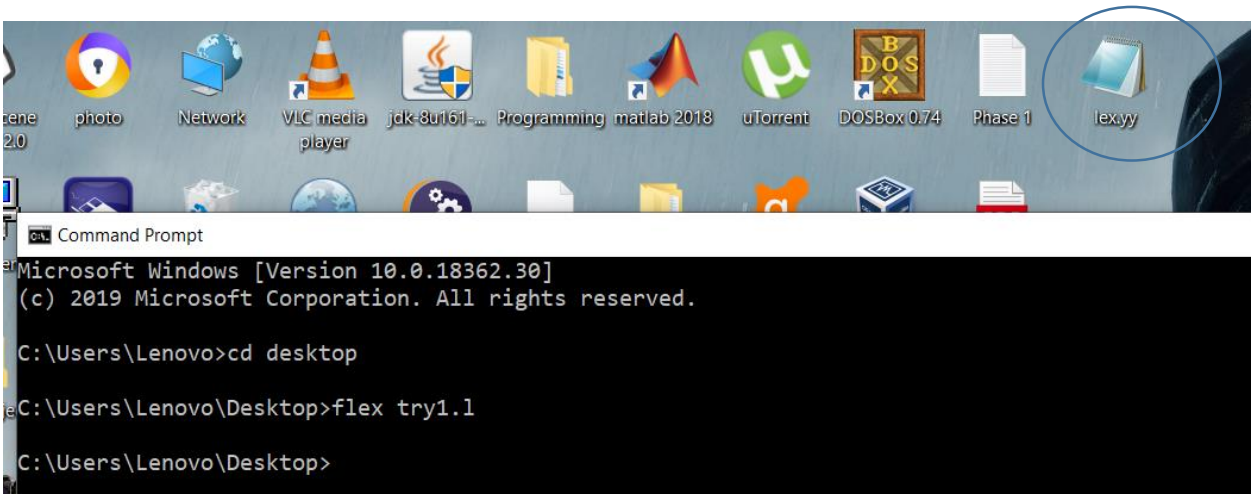
A simple testing code to count the number of occurrences of the letter d lower or upper case and ignore all other letters:

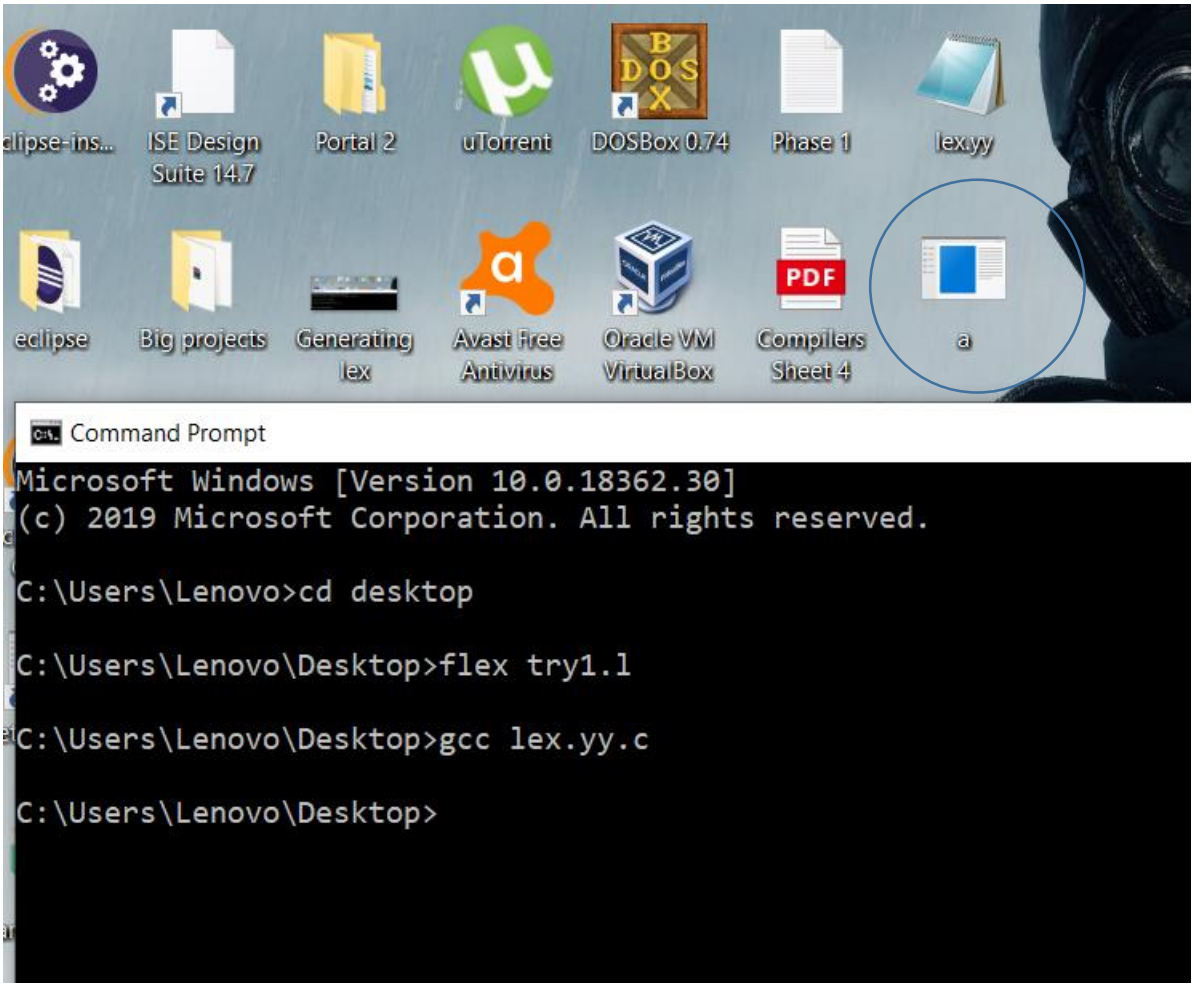
The input file (.l):

(Note: comments if needed are enclosed by /*** ***/ in the .l file format.)

```
%{
 int total_found = 0;
%}

%%
(d|D)  {printf("letter d found\n");
        total_found++;}
.      {printf("%s not d \n", yytext);}
\n     {return 0;}
%%

int yywrap(){}
int main(){

 FILE *fp;
 char filename[50];
 printf("Enter the filename: \n");
 scanf("%s",filename);
 fp = fopen(filename,"r");
 yyin = fp;

yylex();
printf("\nNumber of ds found - %d\n", total_found);

return 0;
}
```

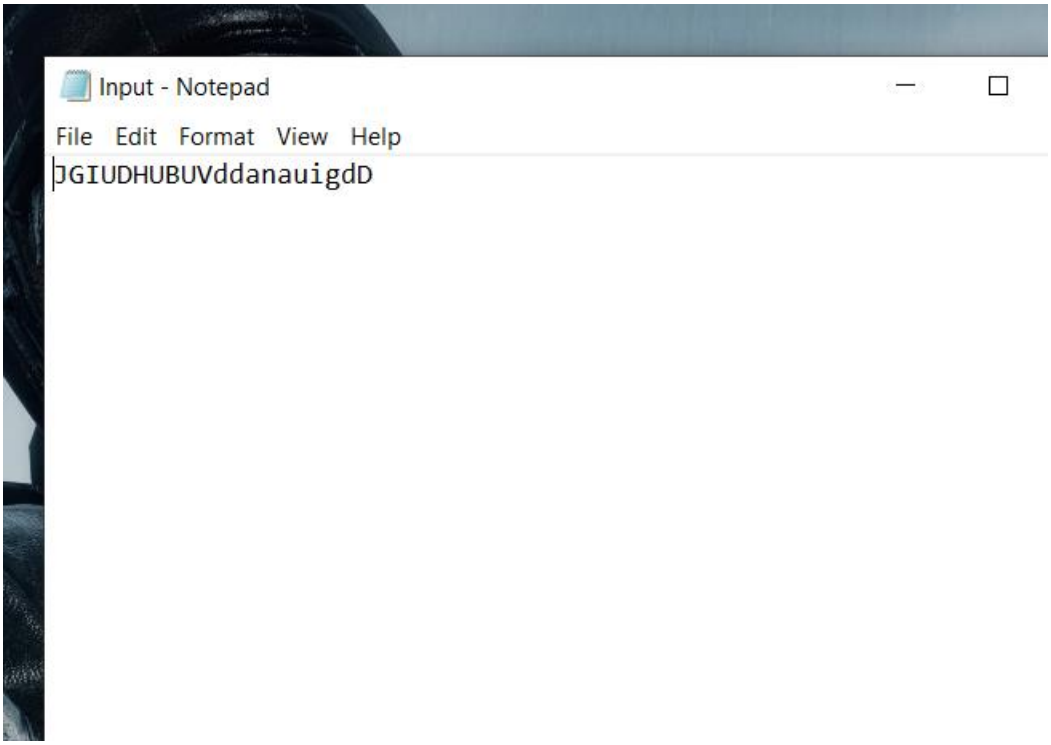
The steps of cmd commands:




clipse-ins...
ISE Design Suite 14.7
Portal 2
uTorrent
DOSBox 0.74
Phase 1
lex.yy
eclipse
Big projects
Generating lex
Avast Free Antivirus
Oracle VM VirtualBox
Compilers Sheet 4
a
Command Prompt
Microsoft Windows [Version 10.0.18362.30]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\Lenovo>cd desktop
C:\Users\Lenovo\Desktop>flex try1.l
C:\Users\Lenovo\Desktop>gcc lex.yy.c
C:\Users\Lenovo\Desktop>

An input file to the scanner:

Has 2 Ds and 3 ds as shown

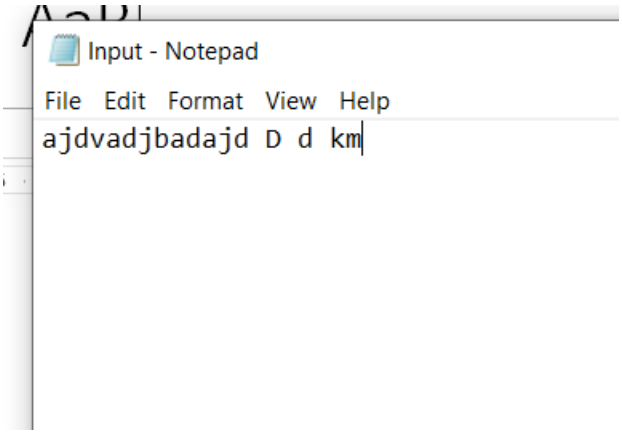


Result of a run:

```
C:\Users\Lenovo\Desktop>a.exe
Enter the filename:
Input.txt
J not d
G not d
I not d
U not d
letter d found
H not d
U not d
B not d
U not d
V not d
letter d found
letter d found
a not d
n not d
a not d
u not d
i not d
g not d
letter d found
letter d found

Number of ds found - 5
```

Another input:

Total is 6



```
C:\Users\Lenovo\Desktop>a.exe
Enter the filename:
Input.txt
a not d
j not d
letter d found
v not d
a not d
letter d found
j not d
b not d
a not d
letter d found
a not d
j not d
letter d found
  not d
letter d found
  not d
letter d found
  not d
k not d
m not d

Number of ds found - 6

C:\Users\Lenovo\Desktop>
```

Second run:

Now let's try the lecture example for lexical analysis:

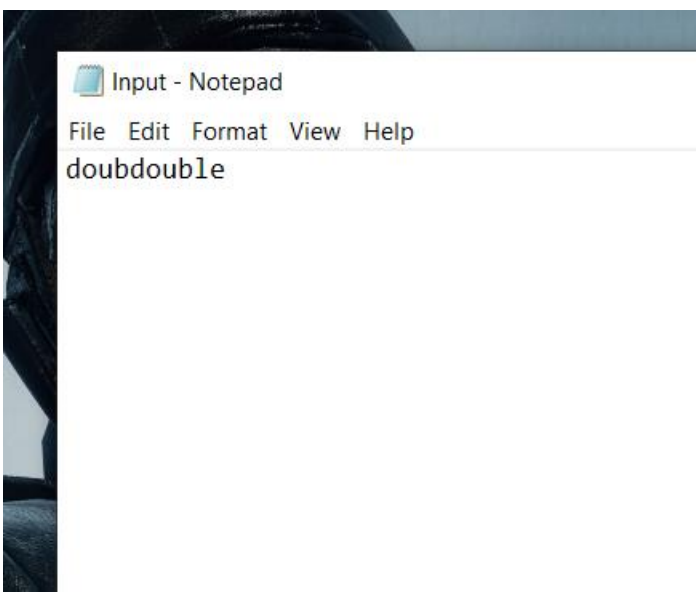
```
T_Do          do
T_Double      double
T_Mystery     [A-Za-z]
```

Here is its implementation:

(lower cases assumed from given expressions)

```
%{
%}

%%
(do) {printf("T_Do \n");}
(double) {printf("T_Double\n");}
[a-zA-Z] {printf("T_Mystery\n");}
.      {printf("%s is unknown \n", yytext);}
\n     {return 0;}
%%

int yywrap(){}
int main(){

 FILE *fp;
 char filename[50];
 printf("Enter the filename: \n");
 scanf("%s",filename);
 fp = fopen(filename,"r");
 yyin = fp;

yylex();
printf("END");

return 0;
}
```

Here is the input and run result:



```
C:\Users\Lenovo\Desktop>a.exe
Enter the filename:
Input.txt
T_Do
T_Mystery
T_Mystery
T_Double
END
C:\Users\Lenovo\Desktop>
```

Third run:

Here we implement the given example in the pdf for JAVA code using the provided expressions and definitions:

The implementation:

Notes: \: escape character, a production is added doing nothing to neglect the space.

```
%{
int found = 0;
%}

%%
(while) {printf("while \n");}
(if) {printf("if \n");}
(else) {printf("else \n");}
(boolean) {printf("boolean \n");}
(int) {printf("int \n");}
(float) {printf("float \n");}
(;) {printf("; \n");}
(,) {printf(", \n");}
\( {printf("( \n");}
\) {printf(") \n");}
\{ {printf("{ \n");}
\} {printf("} \n");}
[a-zA-Z]([a-zA-Z]|[0-9])* {printf("id \n");}
[0-9]+\.[0-9]+(E[0-9]+)?  {printf("num \n"); found = 1;}
[0-9]+ {if(!found)printf("num \n");found = 0;}
(\=\=|!\=|>|<|>\=|<\=) {printf("relop \n");}
(\=) {printf("assign \n");}
(\+|\-) {printf("addop \n");}
(\*|\/) {printf("mulop \n");}
(\ )      {}
.     {printf("%s is error \n", yytext);}
\n    {return 0;}
%%
int yywrap(){}
int main(){

 FILE *fp;
 char filename[50];
 printf("Enter the filename: \n");
 scanf("%s",filename);
 fp = fopen(filename,"r");
 yyin = fp;

yylex();
printf("END");

return 0;
}
```

Input and run:

Input - Notepad

File Edit Format View Help

```
int sum , count , pass , mnt; while (pass != 10)  { pass = pass +
1 ;  }
```

```
C:\Users\Lenovo\Desktop>a.exe
Enter the filename:
Input.txt
int
id
,
id
,
id
,
id
;
while
(
id
relop
num
)
{
id
assign
id
addop
num
;
}
END
C:\Users\Lenovo\Desktop>
```