Practical 1

Name: Rajatkumar Patel
Roll No.: 18BCE191

Aim

To implement lexical analyser to recognize all distinct token classes.

Code

```
%{
	#include<stdio.h>
	int a = 10;
%}
%%
[0-9]   {printf("this is a digit\n");}
[a-z][0-9|a-z]*	{printf("this is an identifier = %s \n",yytext);}
[+|-|%|*|=|//]  {printf("this is an operator, value = %s \n",yytext);}
[>|<|=] {printf("this is a relational operator, value = %s \n",yytext);}
[!|&|>>|<<|~|^] {printf("this is a bitwise operator, value = %s \n",yytext);}
[,|;|(|)|{|}]	{printf("this is a synchronizing token, value = %s \n",yytext);}
\"	{printf("quotes\n");}
[ ]	{printf("this is space\n");}
%%
int yywrap(){}
int main(){
	yylex();
	printf("test program\n");
	return 0;
}
```

Output

```
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals$ ./a.out
hello, my name is Rajat. I am in 7th semester;
this is an identifier = hello
this is a synchronizing token, value = ,
this is space
this is an identifier = my
this is space
this is an identifier = name
this is space
this is an identifier = is
this is space
Rthis is an identifier = ajat
.this is space
Ithis is space
this is an identifier = am
this is space
this is an identifier = in
this is space
this is a digit
this is an identifier = th
this is space
this is an identifier = semester
this is a synchronizing token, value = ;
```

Steps to compile

1. flex Prac1.l
2. gcc lex.yy.c
3. ./a.out

Conclusion

Used flex tool to recognize different tokens from the given input string based on the regex defined. We can use this tool to get different tokens from the given input which can be used as an input to the parser.