**LPCC ASSIGNMENT-3(WEEK-2)**

****

**SUBMITTED TO:**

**PROF. PALLAVI REGE**

**VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**

**COMPUTER ENGINEERING DEPARTMENT**

**BY:**

**NAME: ABHISHEK MORE**

**G.R No.: 21810033**

**ROLL NO.: 323036**

**CLASS: T.Y COMP**

**BATCH: COMP C2**

**ASSIGNMENT-3(WEEK-2)**

**AIM:**

1. Write a program to implement a lexical analyzer for parts of speech, Using LEX.

* 3B -- For parts of speech for subset of ENGLISH language with SYMBOL TABLE
* 3D -- Write Lexical analyser with SYMBOL TABLE for subset of ‘C’ programming language

**SOURCE CODE:**

**3B:**

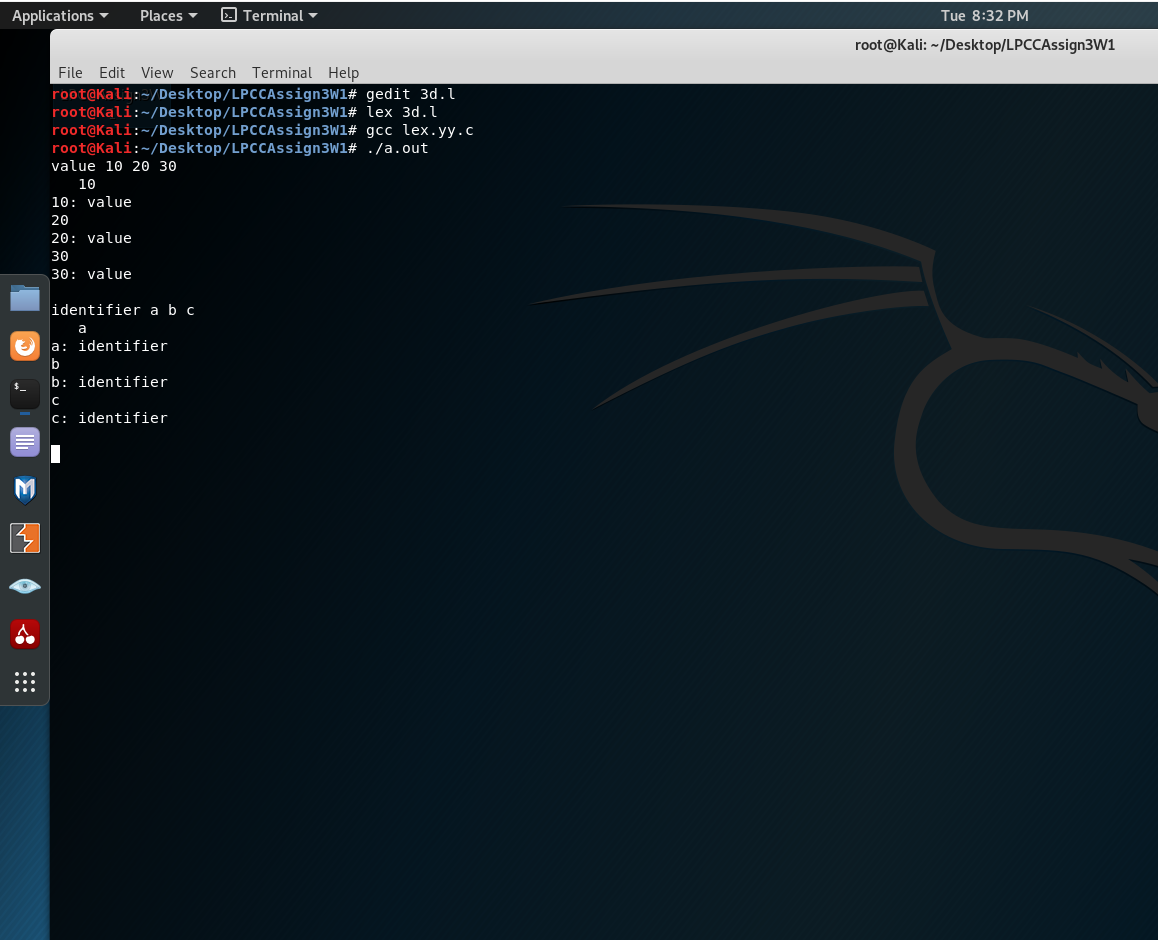
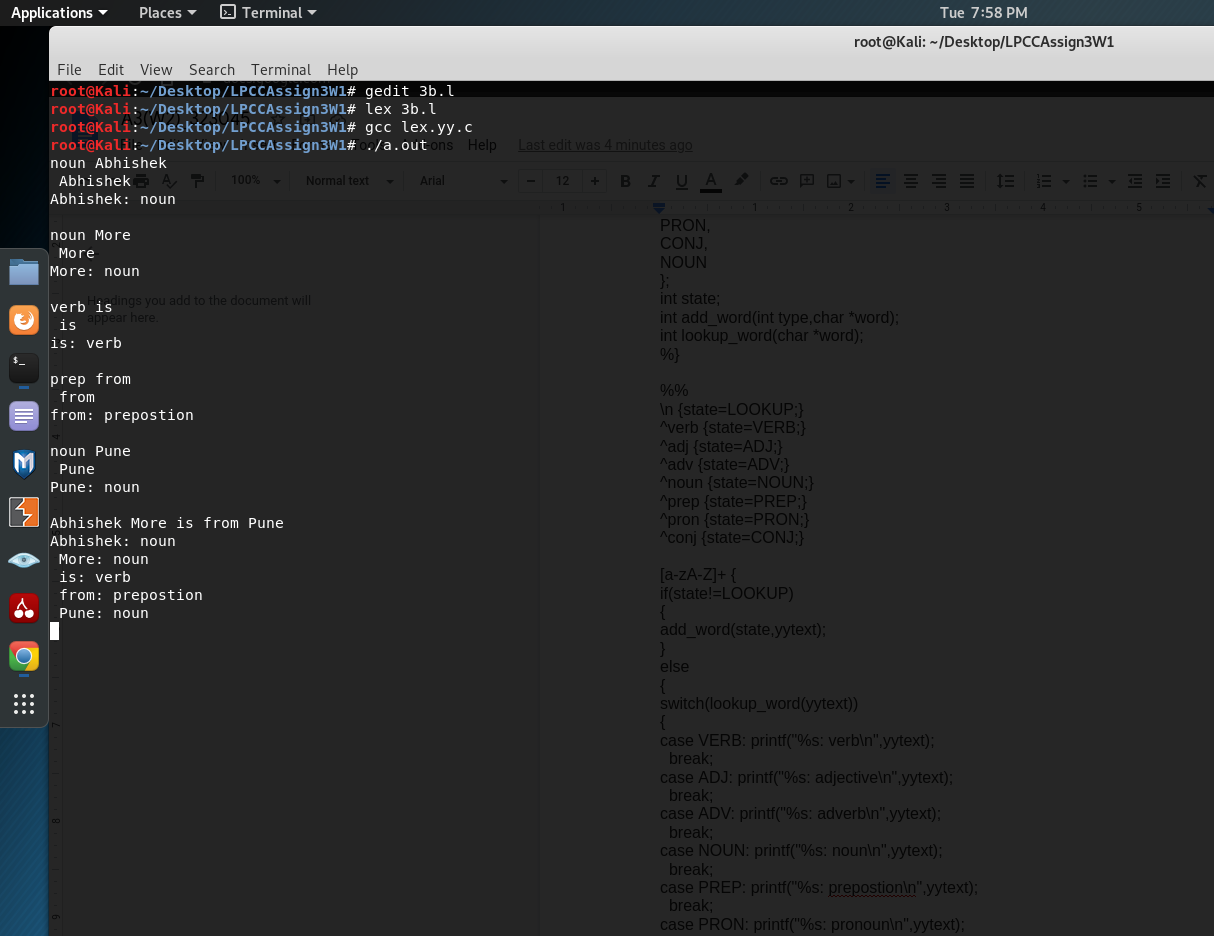
%{  
enum{  
LOOKUP=0,  
VERB,  
ADJ,  
ADV,  
PREP,  
PRON,  
CONJ,  
NOUN  
};  
int state;  
int add\_word(int type,char \*word);  
int lookup\_word(char \*word);  
%}  
  
%%  
\n {state=LOOKUP;}  
^verb {state=VERB;}  
^adj {state=ADJ;}  
^adv {state=ADV;}  
^noun {state=NOUN;}  
^prep {state=PREP;}  
^pron {state=PRON;}  
^conj {state=CONJ;}  
  
[a-zA-Z]+ {  
if(state!=LOOKUP)  
{  
add\_word(state,yytext);  
}  
else  
{  
switch(lookup\_word(yytext))  
{  
case VERB: printf("%s: verb\n",yytext);  
  break;  
case ADJ: printf("%s: adjective\n",yytext);  
  break;  
case ADV: printf("%s: adverb\n",yytext);  
  break;  
case NOUN: printf("%s: noun\n",yytext);  
  break;  
case PREP: printf("%s: prepostion\n",yytext);  
  break;  
case PRON: printf("%s: pronoun\n",yytext);  
  break;  
case CONJ: printf("%s: conjunction\n",yytext);  
  break;  
}  
}  
}  
%%  
  
int main()  
{  
yylex();  
return 0;  
}  
int yywrap()  
{  
return 1;  
}  
  
struct word  
{  
char \*word\_name;  
int word\_type;  
struct word \*next;  
};  
struct word \*word\_list;  
extern void \*malloc();  
int add\_word(int type,char \*word)  
{  
struct word \*wp;  
if(lookup\_word(word)!=LOOKUP)  
{  
printf("Warning : word %s already defined \n",word);  
return 0;  
}  
  
wp=(struct word \*)malloc(sizeof(struct word));  
wp->next=word\_list;  
wp->word\_name=(char\*)malloc(strlen(word)+1);  
strcpy(wp->word\_name,word);  
wp->word\_type=type;  
word\_list=wp;  
return 1;  
}  
  
int lookup\_word(char \*word)  
{  
struct word \*wp=word\_list;  
for(;wp;wp=wp->next)  
{  
  if(strcmp(wp->word\_name,word)==0)  
return wp->word\_type;

}  
}

**3D:**

%{  
enum{  
LOOKUP=0,  
IDENTIFIER,  
RELATIONAL,  
BITWISE,  
LOGICAL,  
ASSIGNMENT,  
VALUE,  
FORMAT  
};  
int state;  
int add\_word(int type,char \*word);  
int lookup\_word(char \*word);  
%}  
  
%%  
\n {state=LOOKUP;}  
^identifier {state=IDENTIFIER;}  
^relational {state=RELATIONAL;}  
^bitwise {state=BITWISE;}  
^logical {state=LOGICAL;}  
^assignment {state=ASSIGNMENT;}  
^value {state=VALUE;}  
^fromat {state=FORMAT;}  
  
  
  
[a-zA-Z0-9\_]\* {  
if(state!=LOOKUP)  
{  
add\_word(state,yytext);  
}  
else  
{  
switch(lookup\_word(yytext))  
{  
case IDENTIFIER: printf("%s: identifier\n",yytext);  
  break;  
case RELATIONAL: printf("%s: relational\n",yytext);  
  break;  
case BITWISE: printf("%s: bitwise\n",yytext);  
  break;  
case LOGICAL: printf("%s: logical\n",yytext);  
  break;  
case ASSIGNMENT: printf("%s: assignment\n",yytext);  
  break;  
case VALUE: printf("%s: value\n",yytext);  
  break;  
case FORMAT: printf("%s: format\n",yytext);  
  break;  
}  
}  
}  
%%  
  
int main()  
{  
yylex();  
return 0;  
}  
int yywrap()  
{  
return 1;  
}  
  
struct word  
{  
char \*word\_name;  
int word\_type;  
struct word \*next;  
};  
struct word \*word\_list;  
extern void \*malloc();  
int add\_word(int type,char \*word)  
{  
struct word \*wp;  
if(lookup\_word(word)!=LOOKUP)  
{  
printf("Warning : word %s already defined \n",word);  
return 0;  
}  
  
wp=(struct word \*)malloc(sizeof(struct word));  
wp->next=word\_list;  
wp->word\_name=(char\*)malloc(strlen(word)+1);  
strcpy(wp->word\_name,word);  
wp->word\_type=type;  
word\_list=wp;  
return 1;  
}  
  
int lookup\_word(char \*word)  
{  
struct word \*wp=word\_list;  
for(;wp;wp=wp->next)  
{  
  if(strcmp(wp->word\_name,word)==0)  
return wp->word\_type;  
}  
}

**OUTPUT:**

**** **3B: 3D:**

**CONCLUSION:**

Successfully implemented a lexical analyser using lex for part of speech with using Symbol table and also implemented lexical analyser for identifying the special characters and operators.