ASSIGNMENT: 03

Name: Abhijeet Biswas
SRN: 201900400
Roll No: 05
Div: B

Question:

Design a Lexical analyzer for the subset of C Language using LEX or FLEX to lookup and also dynamically add new tokens with first word on a line indicating the token class. Upload a single file with input , output and source code.

Input:

keyword void main float printf

void main() { float a = 2, b = 2; float div = a/b; printf (div); }

keyword int

void main() { int a = 5, b = 2; int add = a+b; printf (add); }

Code:

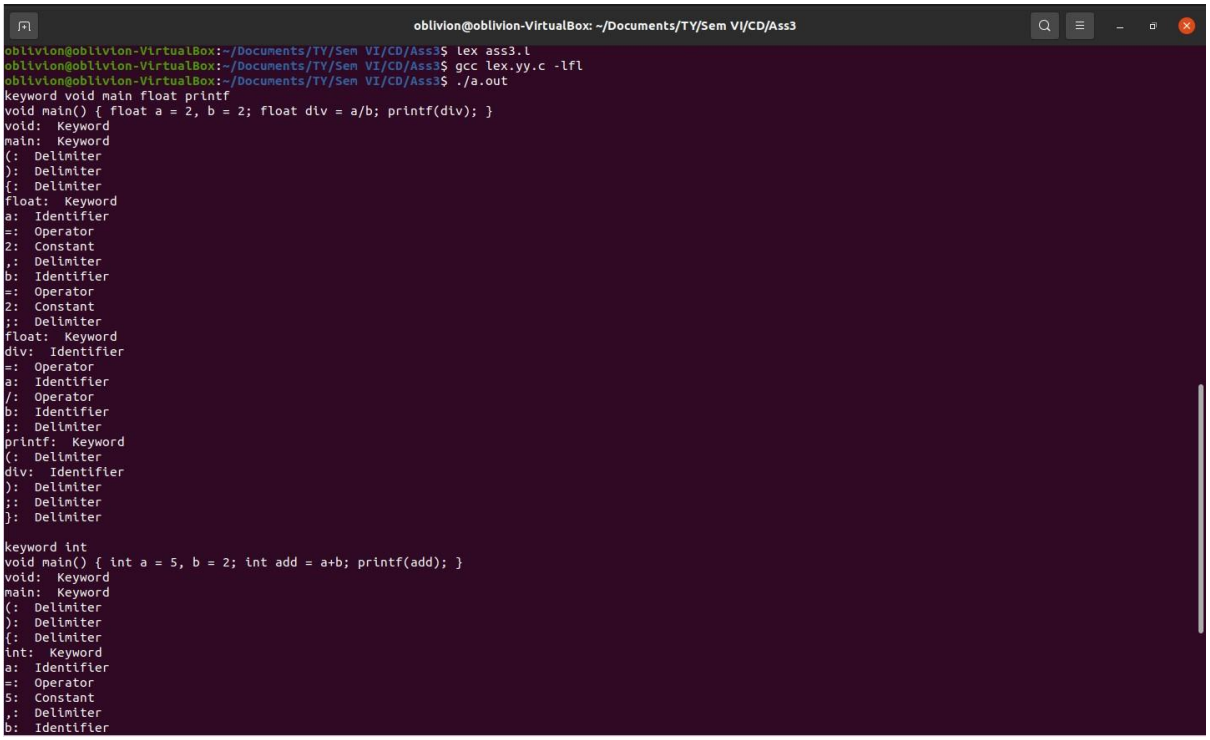
```
%{
enum {
LOOKUP =0, /* default - looking rather than defining. */ keyword,
IDE,
};
int state;
int add_word(int type, char *word); int lookup_word(char *word);
%}
```

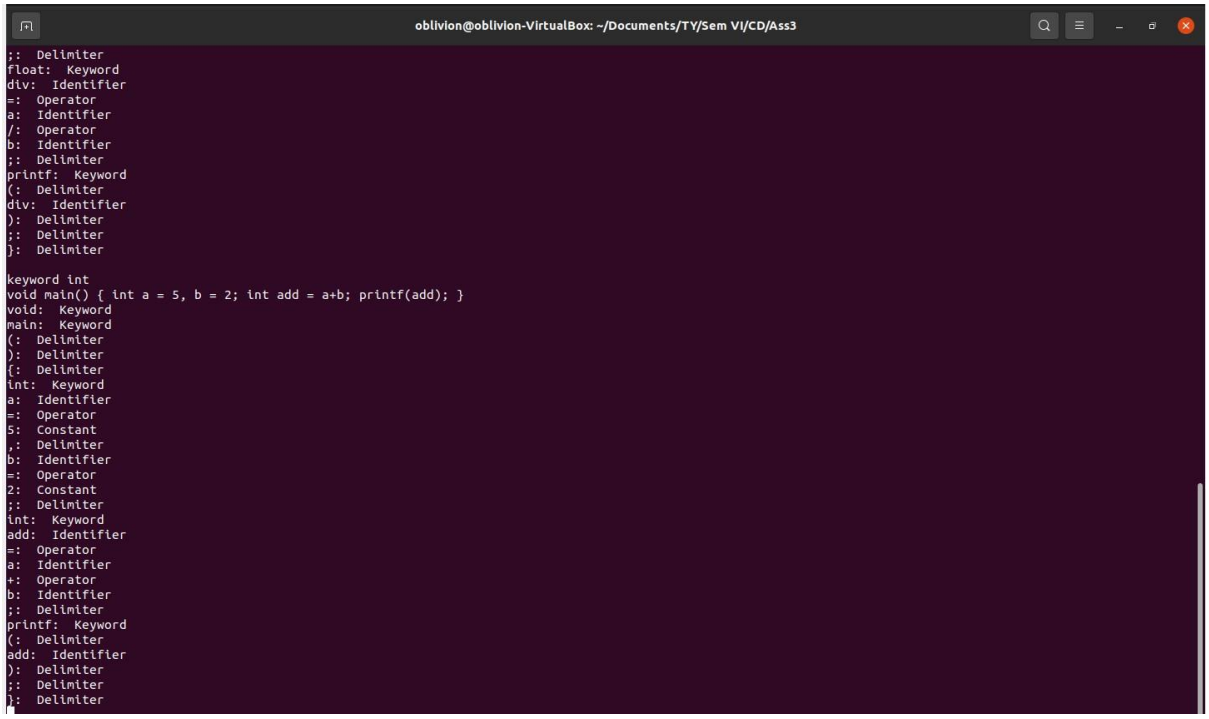
```
%%

\n  { state = LOOKUP; }
^keyword  {  state  =  keyword;  }
^ide  {  state  =  IDE;  }
[0-9]  {  printf("%s:  Constant\n",  yytext);  }

"+"|"-"|"="|"*"|"%"|"/"|"++"|"--"  {  printf("%s:  Operator\n",  yytext);  }

"("|")"|"{"|"}"|"["|"]"|";"|","  {  printf("%s:  Delimiter\n",  yytext);  }

[a-zA-Z]+ {

if(state != LOOKUP) { add_word(state,  yytext);
} else {
switch(lookup_word(yytext))  { case  keyword:
printf("%s:  Keyword\n",  yytext); break;
case  IDE:
printf("%s:  Identifier\n",  yytext); break;
default:
printf("%s:  Identifier\n",  yytext); break;
}
}
}

.  /* ignore anything else */ ;
```

```
%%

int  main()
{
yylex();
}
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;
}
return LOOKUP;
```

}

Output:


oblivion@oblivion-VirtualBox: ~/Documents/TY/Sem VI/CD/Ass3
oblivion@oblivion-VirtualBox:~/Documents/TY/Sem VI/CD/Ass3$ lex ass3.l
oblivion@oblivion-VirtualBox:~/Documents/TY/Sem VI/CD/Ass3$ gcc lex.yy.c -lfl
oblivion@oblivion-VirtualBox:~/Documents/TY/Sem VI/CD/Ass3$ ./a.out
keyword void main float printf
void main() { float a = 2, b = 2; float div = a/b; printf(div); }
void: Keyword
main: Keyword
(: Delimiter
): Delimiter
{: Delimiter
float: Keyword
a: Identifier
=: Operator
2: Constant
,: Delimiter
b: Identifier
=: Operator
2: Constant
;: Delimiter
float: Keyword
div: Identifier
=: Operator
a: Identifier
/: Operator
b: Identifier
;: Delimiter
printf: Keyword
(: Delimiter
div: Identifier
): Delimiter
;: Delimiter
}: Delimiter
keyword int
void main() { int a = 5, b = 2; int add = a+b; printf(add); }
void: Keyword
main: Keyword
(: Delimiter
): Delimiter
{: Delimiter
int: Keyword
a: Identifier
=: Operator
5: Constant
,: Delimiter
b: Identifier


oblivion@oblivion-VirtualBox: ~/Documents/TY/Sem VI/CD/Ass3
;: Delimiter
float: Keyword
div: Identifier
=: Operator
a: Identifier
/: Operator
b: Identifier
;: Delimiter
printf: Keyword
(: Delimiter
div: Identifier
): Delimiter
;: Delimiter
}: Delimiter
keyword int
void main() { int a = 5, b = 2; int add = a+b; printf(add); }
void: Keyword
main: Keyword
(: Delimiter
): Delimiter
{: Delimiter
int: Keyword
a: Identifier
=: Operator
5: Constant
,: Delimiter
b: Identifier
=: Operator
2: Constant
;: Delimiter
int: Keyword
add: Identifier
=: Operator
a: Identifier
+: Operator
b: Identifier
;: Delimiter
printf: Keyword
(: Delimiter
add: Identifier
): Delimiter
;: Delimiter
}: Delimiter