Lab 2: Implement the Lexical analyzer for the given language. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value.

second.l

```
%{
# include<stdio.h>
# include<string.h>
struct symtab
{
char *name;
};
struct symtab sym[10],*k;
struct symtab *install_id(char *s);
void disp();
%}
L    [a-zA-Z]
D    [0-9]
id   {L}({L}|{D})*
num    {D}+(\.{D}+)?([eE][-+]?{D}+)?
bop    [-+*/=]
uop    "++"|"--"
relop          "<"|"<="|">"|">="|"!="|"=="
lop        "&&"|"||"
bitlop       [&|!]
kew   "if"|"else"|"while"|"do"|"for"|"int"|"char"|"float"
pun          [,;'"\[\]\{\}\)\(]
comment        "/\*"(.|\n)*"\*/"|"//"(.)*
ws     [ \t\n]+
st       \"(.)*\"
%%
{ws}    {}
{kew} {printf("keyword=%s\n",yytext);}
{id} {k=install_id(yytext);printf("identifier =%s\n",yytext);}
{num} {printf("constant =%s\n",yytext);}
{bop} {printf("binary op =%s\n",yytext);}
{uop} {printf("unary op =%s\n",yytext);}
{relop} {printf("relational op=%s\n",yytext);}
{lop} {printf("logical op =%s\n",yytext);}
{pun} {printf("punct =%s\n",yytext);}
{bitlop}          {printf("bitwise logical op=%s\n",yytext);}
{comment}     {printf("comment =%s\n",yytext);}
{st}     {printf("string =%s\n",yytext);}
%%
main()
{
```

```
yylex();
disp();
}
struct symtab *install_id(char *s)
{
struct symtab *p;
printf("in symbol table\n");
for(p=&sym[0];p<&sym[10];p++)
{
if(p->name && !strcmp(s,p->name))
return p;
if(!p->name)
{
p->name=strdup(s);
return p;
}
}
```

OUTPUT:

```
-> lex second.l
-> gcc lex.yy.c
second.l:40:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   40 | main()
      | ^~~~
-> ./a.out < inp.c
keyword=int
in symbol table
identifier =main
punct =(
punct =)
punct ={
keyword=int
in symbol table
identifier =a
binary op ==
constant =1
punct =;
in symbol table
identifier =printf
punct =(
string ="Hello world"
punct =)
punct =;
in symbol table
identifier =return
constant =0
punct =;
punct =}
symbol table
main
a
printf
return
->
```