Lab Assignment -4

AP19110010233

NARESH TEJA

1 st code

```
digit [0-9]*
id [a-zA-Z][a-zA-Z0-9]*
num [0-9]\.[0-9]
%{
#include<stdio.h>
#include<string.h>
%}

%%
int |
float |
char |
double |
void { printf("\n %s is keyword",yytext);}
"<=" {printf("\n %s is Relational operator Lessthan or Equal
to",yytext);}
"<" {printf("\n %s is Relational operator Lessthan",yytext);}
```

```
">=" {printf("\n %s is Relational operator Greaterthan or Equal
to",yytext);}
">" {printf("\n %s is Relational operator Greaterthan",yytext);}
"==" {printf("\n %s is Relational operator Equal to",yytext);}
"!=" {printf("\n %s is Relational operator Not Equal to",yytext);}
"=" {printf("\n %s is Assignment operator",yytext);}
{id} {printf("\n %s is identifier",yytext); }
{num} {printf("\n %s is float",yytext);}
{digit} {printf("\n %s is digit",yytext);}
%%
main()
{
yylex();
}

int yywrap()
{
return 1;
}
```

2nd code

```
digit [0-9]*
id [a-zA-Z][a-zA-Z0-9]*
num [0-9]\.[0-9]
%{
#include<stdio.h>
#include<string.h>
int cnt=0,i=0,j=0;
char st[10][10];
int look_up(char st[10][10],char *id,int n);
```

```
%}
%%
int |
float |
char |
double |
void |
main { printf(" \n %s is keyword",yytext);}
{num} { printf("\n %s is float",yytext);}
{id} { printf("\n %s is identifier",yytext);
if (!lookup(st,yytext,i)){
strcpy(st[i++],yytext);cnt++;}
}

{digit} {printf("\n %s is digit",yytext);}
%%
main()
{
yylex();
printf(" No of id are : %d ",cnt);
printf("\n the contents of symbol table are :\n");
for(j=0;j<i;j++)
printf("\n %s",st[j]);

}

int yywrap()
{
return 1;
}
int lookup(char st[10][10],char *id,int n)
{
```

```
for(j=0;j<n;j++)
if(!strcmp(st[j],id))
return 1;
return 0;
}
```

3rd code

```
digit [0-9]*
id [a-zA-Z][a-zA-Z0-9]*
num [0-9]\.[0-9]
%{
#include<stdio.h>
#include<string.h>
int i=0,j=0,cnt=0,n=0,com=0,scom=0;
char st[10][10];
%}
%%
\n {scom=0;n++;}
"//" {scom=1;printf("\n single line comment\n\n");}
"/*" {com=1;printf("\n comment start\n");}
"*/" {com=0;printf("\n comment end\n");}
int |
float |
char |
double |
void {if(!com&&!scom) printf(" \n %s is keyword",yytext);}
"<=" {if (!com&&!scom) printf("\n %s is Relational operator Lessthan or
Equal to",yytext);}
"<" {if(!com&&!scom) printf("\n %s is Relational operator
Lessthan",yytext);}
```

```
">=" {if(!com) printf("\n %s is Relational operator Greaterthan or Equal
to",yytext);}
">" {if(!com&&!scom) printf("\n %s is Relational operator
Greaterthan",yytext);}
"==" {if(!com&&!scom) printf("\n %s is Relational operator Equal
to",yytext);}
"!=" {if (!com&&!scom) printf("\n %s is Relational operator Not Equal
to",yytext);}
{id} {if(!com&&!scom) printf("\n %s is identifier",yytext); }
{num} {if(!com&&!scom) printf("\n %s is float",yytext);}
{digit} {if (!com&&!scom) printf("\n %s is digit",yytext);}
%%

main()
{

}
yywrap()
{
yylex();
printf(" \n\n\n no of lines = %d\n\n",n);
return 0;

return 1;
}
```

4th code

```
_digit [0-9]*
id [a-zA-Z][a-zA-Z0-9]*
num digit*\.digit*
%{
```

```
#include<stdio.h>
#include<string.h>
int cnt=0,i=0,j=0;
char st[10][10];
int look_up(char st[10][10],char *id,int n);
%}
%%
int | float | char | double | void | main { fprintf(yyout," \n %s is
keyword",yytext);}
{num} { fprintf(yyout,"\n %s is float",yytext);}
{id} { fprintf(yyout,"\n %s is identifier",yytext); cnt++;

if(!look_up(st,yytext,i))
strcpy(st[i++],yytext);
}

{digit} {fprintf(yyout,"\n %s is digit",yytext);}
.
%%
main()
{
yyin =fopen("x.txt","r");
yyout=fopen("y.txt","w");
yylex();
printf("\n the contents of symbol table are :\n");
for(j=0;j<i;j++)
printf("\n %s",st[j]); printf("\n\n");
return 0;
}
int yywrap()
{
return 1;
```

```
}
int look_up(char st[10][10],char *id,int n)
{
for(j=0;j<n;j++)
if(!strcmp(st[j],id))
return 1;
return 0;
}
```

5th code

```
digit [0-9]*
id [a-zA-Z][a-zA-Z0-9]*
num digit*\.digit*
%{
#include<stdio.h>
#include<string.h>
int cnt=0,i=0,j=0;
char st[10][10];
int look_up(char st[10][10],char *id,int n);
%}
%%
int |
float |
char |
double |
void { printf(" \n %s is keyword",yytext);}
{num} { printf("\n %s is float",yytext);}
{id} { printf("\n %s is identifier",yytext);

cnt++;
if(!look_up(st,yytext,i))
```
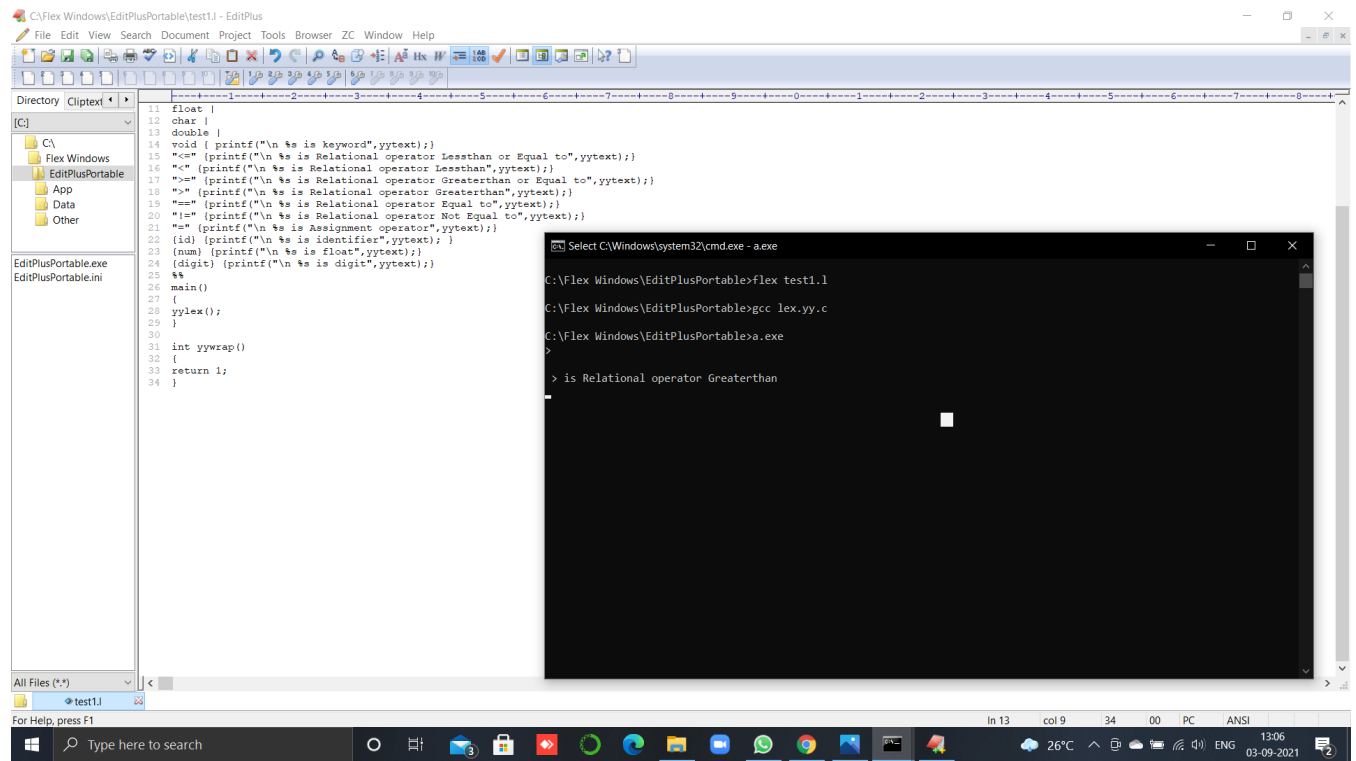
```
strcpy(st[i++],yytext);
}

{digit} {printf("\n %s is digit",yytext);}
%%
main(int argc, char **argv) {
yyin=fopen(argv[1],"r"); // passing input file name as argv[1]
yylex();
return 0;
}
int yywrap()
{
return 1;
}
int look_up(char st[10][10],char *id,int n)
{
for(j=0;j<n;j++)
if(!strcmp(st[j],id))
return 1;
return 0;
}
```
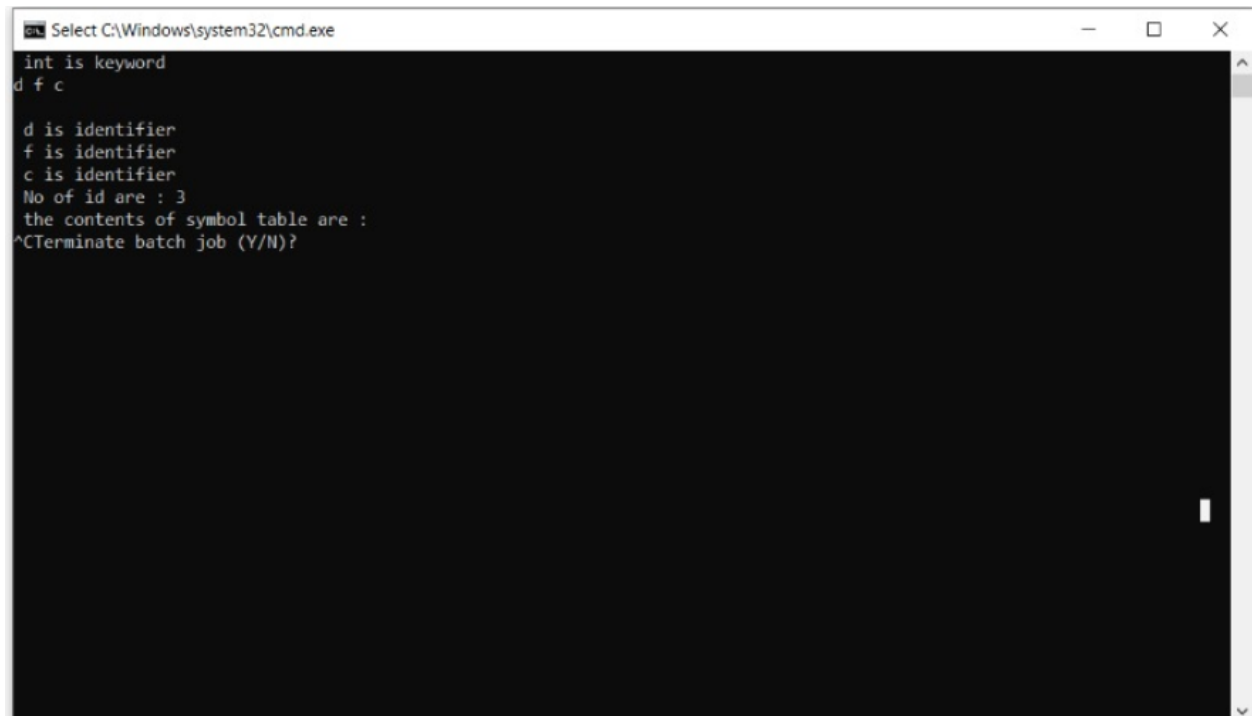
Outputs

Output 1 -

The EditPlus window shows a Flex/Lex source file `test1.l` with the following visible content:

```
11  float |
12  char |
13  double |
14  void { printf("\n %s is keyword",yytext);}
15  "<=" {printf("\n %s is Relational operator Lessthan or Equal to",yytext);}
16  "<" {printf("\n %s is Relational operator Lessthan",yytext);}
17  ">=" {printf("\n %s is Relational operator Greaterthan or Equal to",yytext);}
18  ">" {printf("\n %s is Relational operator Greaterthan",yytext);}
19  "==" {printf("\n %s is Relational operator Equal to",yytext);}
20  "!=" {printf("\n %s is Relational operator Not Equal to",yytext);}
21  "=" {printf("\n %s is Assignment operator",yytext);}
22  {id} {printf("\n %s is identifier",yytext); }
23  {num} {printf("\n %s is float",yytext);}
24  {digit} {printf("\n %s is digit",yytext);}
25  %%
26  main()
27  {
28  yylex();
29  }
30
31  int yywrap()
32  {
33  return 1;
34  }
```

The command prompt window shows:

```
C:\Flex Windows\EditPlusPortable>flex test1.l

C:\Flex Windows\EditPlusPortable>gcc lex.yy.c

C:\Flex Windows\EditPlusPortable>a.exe
>
> is Relational operator Greaterthan
```

Output 2 -

```
int is keyword
d f c

 d is identifier
 f is identifier
 c is identifier
No of id are : 3
 the contents of symbol table are :
^CTerminate batch job (Y/N)?
```

Output3 -

C:\Windows\system32\cmd.exe

```
D:\Work\lex\test3
aebg123i
123

no of vowel=3
no of consonant=2Terminate batch job (Y/N)?
```

Output4 -

```
C:\Windows\system32\cmd.exe

D:\Work\lex\test4
12300
string accepted
3210
string rejected
```

Output 5   -

```
C:\Windows\system32\cmd.exe

D:\Work\lex\test5
2300043
string accepted
2300
string rejected
23000
string accepted
```