CODE

compound.y

```
%{
   #include<stdio.h>
   extern int yylex();
   extern int yywrap();
   extern int yyparse();
%}

%token WH IF OP CP CMP SC ASG ID NUM OPR

%%
start: swh | sif;
swh: WH OP cmpn CP stmt            {printf("VALID SINGLE STATEMENT WHILE\n");}
sif: IF OP cmpn CP stmt            {printf("VALID SINGLE STATEMENT IF\n");}
cmpn: ID CMP ID | ID CMP NUM;
stlst:stmt stlst | stmt;
stmt: ID ASG ID OPR ID SC | ID ASG ID OPR NUM SC
     | start                       {printf("NESTED INSIDE A");}
     ;

%%
int yyerror(char *str)
{
   printf("%s",str);
}
int main()
{
   yyparse();
}
```

compound.l

```
%{
   #include<stdio.h>
   #include<stdlib.h>
   #include "y.tab.h"
   extern int yyerror(char *str);
   extern int yyparse();
%}

%%
```

```
"while" return WH;
"if" return IF;
"(" return OP;
")" return CP;
"<" |
">" |
"<=" |
">=" |
"==" |
"!=" return CMP;
"+" |
"-" |
"*" |
"/" return OPR;
"=" return ASG;
([a-zA-Z])("_"|[a-zA-Z0-9])* return ID;
[0-9]+ return NUM;
";" return SC;
" " {}

%%

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

OUTPUT

```
➜  Lab 7 git:(master) ✗ flex compound.l
➜  Lab 7 git:(master) ✗ yacc -d compound.y
compound.y: warning: 1 nonterminal useless in grammar [-Wother]
compound.y: warning: 2 rules useless in grammar [-Wother]
compound.y:15.1-5: warning: nonterminal useless in grammar: stlst [-Wother]
   15 | stlst:stmt stlst | stmt;
      | ^~~~~
➜  Lab 7 git:(master) ✗ gcc lex.yy.c y.tab.c
y.tab.c: In function ‘yyparse’:
y.tab.c:1391:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’?
[-Wimplicit-function-declaration]
 1391 |       yyerror (YY_("syntax error"));
```

```
    |       ^~~~~~
    |       yyerrok
➜  Lab 7 git:(master) ✗ ./a.out
while(a<b)a=a+v;
VALID SINGLE STATEMENT WHILE

^C
➜  Lab 7 git:(master) ✗ ./a.out
if(a<5) a=c+15;
VALID SINGLE STATEMENT IF

^C
```