11.l

```
%{
#include "y.tab.h"
#include <stdio.h>
#include <stdlib.h>
%}

%%
"=" { return EQUALS; }
"*" { return ASTERISK; }
[a-zA-Z][a-zA-Z0-9]* { yylval = yylval = strdup(yytext); return ID; }
[ \t\n] ; /* Ignore whitespace and newline */
. { return yytext[0]; }
%%

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

12.l

```
%{
#include "y.tab.h"
#include <stdio.h>
#include <stdlib.h>
%}

%%

"int" { return INT; }
"float" { return FLOAT; }
[a-zA-Z][a-zA-Z0-9]* { yylval = strdup(yytext); return ID; }
"," { return ','; }
[ \t\n] ; /* Ignore whitespace and newline */
. { return yytext[0]; }
%%

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

11.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yylex();
void yyerror(const char* s);

extern int yylval;

%}

%token ID EQUALS ASTERISK

%%
S : L EQUALS R { printf("Valid: Assignment\n"); }
| R { printf("Valid: Expression\n"); }
;

L : ASTERISK R { printf("Valid: Pointer\n"); }
| ID { printf("Valid: Variable\n"); }
;

R : L { /* Empty action */ }
;


%%

void yyerror(const char* s) {
fprintf(stderr, "Error: %s\n", s);
}

int main() {
yyparse();
return 0;
}
```

12.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yylex();
void yyerror(const char* s);

extern int yylval;

%}

%token INT FLOAT ID

%%

D : T L { printf("Valid: Declaration\n"); }
;

T : INT { printf("Valid: Type - int\n"); }
| FLOAT { printf("Valid: Type - float\n"); }
;

L : L "," ID { printf("Valid: Variable\n"); }
| ID { printf("Valid: Variable\n"); }
;

%%

void yyerror(const char* s) {
fprintf(stderr, "Error: %s\n", s);
}

int main() {
yyparse();
return 0;
```