

CSB 353: Compiler Design

LAB 9

Submitted By:

Name: PREM KUMAR

Roll No: 191210037

Branch: CSE

Semester: 6 th

Submitted To: Dr. Shelly Sachdeva

Department of Computer Science and Engineering



NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2019-2023

Ques 1. Design a grammar for a declarative statement for C program. Further, write a Yacc program to check if entered statement is a valid declarative statement according to the grammar generated.

Code:

- prog1.l (Lex File)

```
lab9 > ≡ prog1.l
1  %{
2  #include "y.tab.h"
3  #include<stdio.h>
4  int yylval;
5  %}
6  %%
7  "int"[ ]+ {return KEYWORD;}
8  "float"[ ]+ {return KEYWORD;}
9  "char"[ ]+ {return KEYWORD;}
10 [a-zA-Z][a-zA-Z0-9]* {return ID;}
11 [0-9]+ {return NUMBER;}
12 [ \t] ;
13 [,] {return COMMA;}
14 [;] {return SEMICOLON;}
15 \n {return 0;}
16 . {return yytext[0];}
17 %%
18 int yywrap()
19 {
20     return 1;
21 }
22
```

- prog1.y (Yacc File)

```
≡ prog1.y  ×
lab9 > ≡ prog1.y
1  %{
2  #include<stdio.h>
3  int yylex();
4  int yyerror(char* s);
5  int flag=0;
6  %}
7  %token ID KEYWORD SEMICOLON COMMA NUMBER
8
9  %%
10
11  stmt: KEYWORD list SEMICOLON {printf("\nDeclaration statement is valid");} ;
12  list: ID COMMA list | ID ;
13
14  %%
15  int main()
16  {
17  printf("Enter valid declaration statement\n");
18  yyparse();
19  }
20  int yyerror(char* s)
21  {
22  printf("Invalid declaration statement\n");
23  }
24
```

Output:

```
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> flex .\prog1.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> bison -dy .\prog1.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid declaration statement
int a;

Declaration statement is valid
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid declaration statement
int a,b;

Declaration statement is valid
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid declaration statement
array a;
Invalid declaration statement
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> □
```

Ques 2. Design a grammar for a relational expression of C language. Further, write a Yacc program to check if entered statement is a valid relational expression according to the grammar generated.

Code:

- prog2.l (Lex File)

```
≡ prog2.l ×
lab9 > ≡ prog2.l
1  %{
2  #include "y.tab.h"
3  #include<stdio.h>
4  int yylval;
5  %}
6  %%
7  [a-zA-Z][a-zA-Z0-9]* {return ID;}
8  [0-9]+ {return NUM;}
9  "<"| "<="| ">"| ">="| "!="| "==" {return RELATIONAL_OPERATOR;}
10 "(" {return OPENING_PARENTHESIS;}
11 ")" {return CLOSING_PARENTHESIS;}
12 [ \t] ;
13 \n {return 0;}
14 . {return yytext[0];}
15 %%
16 int yywrap(){
17 return 1;
18 }
19
```

- prog2.y (Yacc File)

```
lab9 > prog2.y
1  %{
2  #include<stdio.h>
3  int yylex();
4  int yyerror(char* s);
5  %}
6  %token ID NUM OPENING_PARANTHESIS CLOSING_PARANTHESIS
7  %token RELATIONAL_OPERATOR
8  %left RELATIONAL_OPERATOR
9  %%
10 RelationalExpression : E {
11     printf("Valid Relational Expression\n\n");
12     return 0;
13 }
14 E : expr RELATIONAL_OPERATOR expr | OPENING_PARANTHESIS E CLOSING_PARANTHESIS
15 expr : ID | NUM
16 %%
17
18 int main(){
19     printf("Enter expression:\n");
20     yyparse();
21 }
22
23 int yyerror(char* s) {
24     printf("Invalid Relational Expression\n\n");
25 }
26
```

Output:

```
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> flex .\prog2.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> bison -dy .\prog2.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a>b
Valid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a!=b
Valid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a||b
Invalid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a&&b
Invalid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> □
```

Ques 3. Design a grammar for a logical expression of C language. Further, write a Yacc program to check if entered statement is a valid logical expression according to the grammar generated.

Code:

- prog3.l (Lex File)

```
≡ prog3.l  X
lab9 > ≡ prog3.l
1  %{
2  #include "y.tab.h"
3  #include<stdio.h>
4  int yylval;
5  %}
6  %%
7  "&&"|"||" {return LOGICAL_OPERATOR;}
8  "(" {return OPENING_PARANTHESIS;}
9  ")" {return CLOSING_PARANTHESIS;}
10 [a-zA-Z][a-zA-Z0-9]* {return ID;}
11 [0-9]+ {return NUM;}
12 [ \t] ;
13 \n {return 0;}
14 . {return yytext[0];}
15 %%
16 int yywrap(){
17 return 1;
18 }
19
```


- prog3.y (Yacc File)

```
lab9 > prog3.y
1  %{
2  #include<stdio.h>
3  int yylex();
4  int yyerror(char* s);
5  %}
6
7  %token ID NUM OPENING_PARANTHESIS CLOSING_PARANTHESIS
8  %token LOGICAL_OPERATOR
9  %left LOGICAL_OPERATOR
10 %%
11 LogicalExpression : E {
12     printf("Valid logical expression\n\n");
13     return 0;
14 }
15 E : expr LOGICAL_OPERATOR expr | OPENING_PARANTHESIS E CLOSING_PARANTHESIS
16 expr : ID | NUM
17 %%
18 int main()
19 {
20     printf("Enter valid logical expression\n");
21     yyparse();
22 }
23 int yyerror(char* s)
24 {
25     printf("Invalid logical expression\n\n");
26 }
27
```

Output:

```
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> flex .\prog3.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> bison -dy .\prog3.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid logical expression
a||b
Valid logical expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid logical expression
a&&b
Valid logical expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid logical expression
a!!b
Invalid logical expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> □
```