

Date: 23/10/2023

Roll No. and Name: Yash Ginoya (20BCE075)

Course Code and Name: Compiler Construction (2CS701)

Practical No.: 7

Aim:

To implement grammar rules for control statements, and Loopcontrol

Input Files:

- 7.y

```
% {
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
% }
```

```
%token ID NUM IF '{' '}' LE GE EQ NE OR AND ELSE ELSEIF  
FOR
```

```
%right '='
```

```
%left AND OR
```

```
%left '<' '>' LE GE EQ NE
```

```
%left '+' '-'
```

```
%left '*' '/'
```

```
%left '!'
```

```
% %
```

```
Z:Ifelse|S;
```

```
Ifelse: ST{
```

```
    printf("Valid If else Statement\n");
```

```
    return 0;
```

```
};
```

```
S: A {
```

```
    printf("Valid for loop Statement.\n");exit(0);
```

```
};
```

```
ST:IF '(' E2 ')' '{' ST1';"}' ELSE '{' ST1';"}'
```

```

|IF '(' E2 ')' '{' ST1';"}' G ELSE '{' ST1';"}'
|IF '(' E2 ')' '{' ST1';"}'
|IF '(' E2 ')' '{' '}'

```

```

G:ELSEIF '(' E2 ')' '{' ST1';"}'
|ELSEIF '(' E2 ')' '{' ST1';"}' G

```

```

A: FOR '(' E ';' E2 ';' E ')' B;

```

```

B: '{'
D '}'
|E';'
|A
|
;

```

```

D:D D
|E';'
|A
|
;

```

```

ST1:ST
| E
|
;

```

```

E : ID='E
| E+'E
| E-'E
| E'*E
| E/'E
| E '<'E
| E '>'E

```

```
| E LE E
| E GE E
| E EQ E
| E NE E
| E OR E
| E AND E
| ID
| NUM
| E'+'+'
| E'-'-'
;
```

```
E2 : E'<'E
| E'>'E
| E LE E
| E GE E
| E EQ E
| E NE E
| E OR E
| E AND E
| ID
| NUM
;
```

%%

```
void main()
{
    printf("Enter the exp : ");
    yyparse();
}
void yyerror(){
    printf("\nEntered Statement is Invalid\n\n");
}
```

```
//if(x==y){z=1;}elseif(r==x){q=1;}else{e=1;}
```

- **7.1**

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

```
% }
```

```
alpha [A-Za-z]
```

```
digit [0-9]
```

```
% %
```

```
[ \t\n]
```

```
if return IF;
```

```
else return ELSE;
```

```
elseif return ELSEIF;
```

```
for return FOR;
```

```
{digit}+ return NUM;
```

```
{alpha}({alpha}|{digit})* return ID;
```

```
"{" return '{';
```

```
"}" return '}';
```

```
"<=" return LE;
```

```
">=" return GE;
```

```
"==" return EQ;
```

```
"!=" return NE;
```

```
"||" return OR;
```

```
"&&" return AND;
```

```
. return yytext[0];
```

```
% %
```

```
int yywrap()
```

```
{
```

```
return 1;
```

```
}'
```

- **Output:**

```
nirma@nirma-27: ~/Desktop/cc_7
(base) nirma@nirma-27:~/Desktop/cc_7$ lex 7.l
(base) nirma@nirma-27:~/Desktop/cc_7$ yacc -d 7.y
7.y: warning: 13 shift/reduce conflicts [-Wconflicts-sr]
7.y: warning: 4 reduce/reduce conflicts [-Wconflicts-rr]
(base) nirma@nirma-27:~/Desktop/cc_7$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-27:~/Desktop/cc_7$ ./a.out
Enter the exp : if(x==y){z=1;}elseif(r==x){q=1;}else{e=1;}
Valid If else Statement
(base) nirma@nirma-27:~/Desktop/cc_7$
```

```
nirma@nirma-27: ~/Desktop/cc_7
(base) nirma@nirma-27:~/Desktop/cc_7$ lex 7.l
(base) nirma@nirma-27:~/Desktop/cc_7$ yacc -d 7.y
7.y: warning: 13 shift/reduce conflicts [-Wconflicts-sr]
7.y: warning: 4 reduce/reduce conflicts [-Wconflicts-rr]
(base) nirma@nirma-27:~/Desktop/cc_7$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-27:~/Desktop/cc_7$ ./a.out
Enter the exp : if(x==y){z=1;}elseif(r==x){q=1;}else{e=1;}
Valid If else Statement
(base) nirma@nirma-27:~/Desktop/cc_7$ ./a.out
Enter the exp : for(i=0;i<n;i++){
Valid for loop Statement.
(base) nirma@nirma-27:~/Desktop/cc_7$
```

Conclusion:

After performing this practical, I learn how to implement grammar rules for control statements, and Loopcontrol
