



**NOTRE DAME UNIVERSITY**  
**BANGLADESH**

**CSE-3104 LAB Report-05**

**Course Title: Compiler Design Lab**

**Course Code: CSE-3104**

**Submitted by:**

**Name: Istiak Alam**

**ID: 0692230005101005**

**Batch: CSE-20**

**Submission Date: 24-11-24**

**Lab Task Topic: Compiler Problem Solving using Lex**

**Submitted to:**

**Khorshed Alam**

**Lecturer, NDUB**

## **Problem:**

Write a YACC Program to implement for-loop.

## **Solution:**

### **Code : –**

```
// Lex File -> forloop.l //
%{
#include "y.tab.h"
#include <stdlib.h>
%}

alpha [a-zA-Z]
digit [0-9]

%%
[\\t\\n]
for return FOR;
{digit}+ return NUM;
{alpha}({alpha}|{digit})* return ID;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
"||" return OR;
"&&" return AND;
. return yytext[0];
%%

int yywrap()
{
    return 0;
}

// Yacc File -> forloop.y //
%{
#include <stdio.h>
#include <stdlib.h>

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

%token ID FOR NUM LE GE EQ NE OR AND
%right '=' UMINUS
%left AND OR '<'>' LE GE EQ NE '+' '-' '*' '/' '!'

%%
S:
ST { printf("For Loop Statement Input Accepted.\\n");
    exit(0); }
;

ST:
FOR '(' E ';' E2 ';' E ')' DEF;
DEF: '{' BODY '}'
| E ';'
| ST
|
;

BODY:
```

```

BODY BODY
|E ';'
|ST
|
;

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
|E '+' '+'
|E '-' '-'
|ID
|NUM
;

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
;
%%

int main()
{
  printf("Output");
  yyparse();
  return 0;
}

```

## Code Processing –

Step 1 : Open terminal in Linux and create a lex file named **forloop.l**

⇒ **touch forloop.l**

Step 1 : Open terminal in Linux and create a yacc file named **forloop.y**

⇒ **touch forloop.y**

Step 2 : After Writing those code save it and type command in terminal :

⇒ **bison -d forloop.y**

⇒ **lex forloop.l**

Step 3 : It will create a lex.yy.c & yyfile. Then next type command in terminal

⇒ **gcc -o ForLoop forloop.tab.c lex.yy.c -lfl -lm**

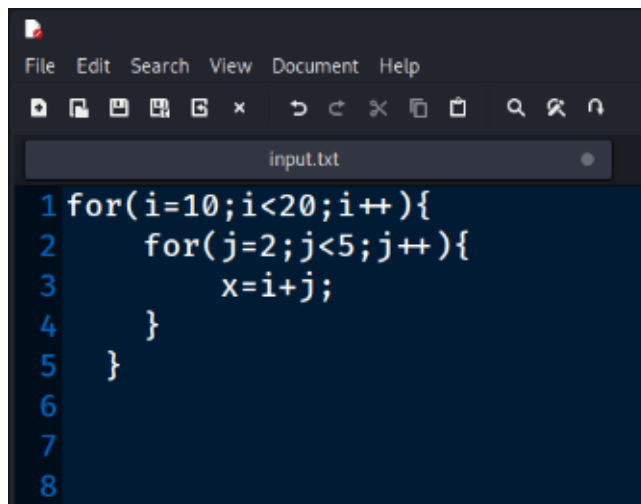
Step 4 : Then it will create an executable file named **ForLoop.exe**.

Step 5 : Next type command in terminal and run the exe file :

⇒ **./ForLoop < input.txt**

## Input and Output:

1. We are using **input.txt** file for checking out inputs in this lexical analyzer.
2. In the **input.txt** file there are a simple program language –



```
File Edit Search View Document Help
input.txt
1 for(i=10;i<20;i++){
2     for(j=2;j<5;j++){
3         x=i+j;
4     }
5 }
6
7
8
```

3. After injecting the input, the output is –



```
kali@kali
File Actions Edit View Help
(kali@kali)-[~/Documents/For-Loop]
└─$ lex forloop.l
(kali@kali)-[~/Documents/For-Loop]
└─$ bison -d forloop.y
forloop.y: warning: 13 shift/reduce conflicts [-Wconflicts-sr]
forloop.y: warning: 4 reduce/reduce conflicts [-Wconflicts-rr]
forloop.y: note: rerun with option '-Wcounterexamples' to gener
(kali@kali)-[~/Documents/For-Loop]
└─$ gcc -o ForLoop y.tab.c lex.yy.c -lfl -lm
(kali@kali)-[~/Documents/For-Loop]
└─$ ./ForLoop
Enter Expression :
for(i=10;i<20;i++)
{
for(j=2;j<5;j++)
{
x=i+j;
}
}
Input Accepted.
(kali@kali)-[~/Documents/For-Loop]
└─$
```