BCSE307P – Compiler Design Lab

Winter Semester 2023-24

Assessment 4

Working with lex and yacc tools

Name: Sujay Ghosh

Reg. No: 21BLC1607

Slot: L7 + L8

Faculty: Dr. Rathna

Task 1:

Lex and Yacc program for expression validation

Code:

Lex:

```
%{
#include "y.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z0-9]* {return ID;}
[0-9]+ {return NUM;}
[\t] {;}
[\n] {return 0;}
. {return yytext[0];}
%%
int yywrap(void)
{
{return 1;}
}
```

Yacc:

```
%{
#include<stdio.h>
#include<stdlib.h>
int n = 2;
void yyerror();
int yylex();
%}


%start stmt
%token ID NUM
%left '+' '-'
%left '/' '*'
%%

stmt : expr
    | ID '=' expr
;
expr : expr '+' expr
       | expr '-' expr
       | expr '*' expr
       | expr '/' expr
       | '(' expr ')'
       | ID
       | NUM
;
%%

void main()
{
```

```
        while(n)
        {
                printf("INPUT AN EXPRESSION : ");
                yyparse();
                printf("VALID EXPRESSION IDENTIFIED\n\n");
        }
}


void yyerror()
{
        printf("EXPRESSION IS INVALID\n\n");
        exit(0);
}
```

Output:

```
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ yacc -d bas1.y
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ lex bas1.l
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ gcc lex.yy.c y.tab.c -o ./bas1
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ ./bas1
INPUT AN EXPRESSION : a+(b-c)
VALID EXPRESSION IDENTIFIED

INPUT AN EXPRESSION : a-(b*c
EXPRESSION IS INVALID
```

Task 2:

Lex and Yacc program for identifier validation

Code:

Lex:

```
%{

#include "y.tab.h"

%}



%%

[a-zA-Z_][_a-zA-Z0-9]* {return ID;}

[\t] {;}
```

```
[\n] {return 0;}

. {return yytext[0];}

%%

int yywrap(void)

{

      return 1;

}
```

Yacc:

```
%{

      #include<stdlib.h>

      #include<stdio.h>

      void yyerror();

      int yylex();

      int n = 1;

%}



      %start stmt

      %token ID



      %%

      stmt : ID

      ;

      %%

      void main()
```

```
{
        while(n)
        {
        printf("INPUT AN IDENTIFIER : ");
        yyparse();
        printf("VALID IDENTIFIER\n\n");
        }
}
void yyerror()
{
        printf("INVALID IDENTIFIER\n\n");
        exit(0);
}
```

Output:

```
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ yacc -d bas2.y
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ lex bas2.l
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ gcc lex.yy.c y.tab.c -o ./bas2
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ ./bas2
INPUT AN IDENTIFIER : name
VALID IDENTIFIER

INPUT AN IDENTIFIER : 3dimension
INVALID IDENTIFIER

master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ ./bas2
INPUT AN IDENTIFIER : _name
VALID IDENTIFIER

INPUT AN IDENTIFIER : $dollar
INVALID IDENTIFIER

master@master-VirtualBox:~/6th sem/cd_lab/lab 4$
```

Task 3:

Lex and Yacc program for evaluating expression

Code:

Lex:

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


        %%
        "print"     {return print;}
        "exit"      {return end;}
        [a-zA-Z]     {yylval.id = yytext[0]; return identifier;}
        [0-9]+       {yylval.n = atoi(yytext); return num;}
        [\t\n]       ;
        [-=+/;()]      {return yytext[0];}
        "*"             {return yytext[0];}
        .               {ECHO; yyerror();}
        %%
```

Yacc:

```
%{
        #include<stdio.h>
        #include<stdlib.h>
        void yyerror();
        int sym[52];
        int value(char c);
        void update(char s,int val);
%}

%union{int n; char id;}
%start stmt
%token print end
%token <n> num
%token <id> identifier
%type <n> stmt exp term
%type <idnt> assign
%right '='
%left '+' '-'
%left '*' '/'
%%

stmt : assign ';'                                        {;}
        | end ';'
        {exit(EXIT_SUCCESS);}
        | print exp ';'
        {printf("\nValue is : %d\n",$2);}
        | stmt assign ';'                                {;}
```

```
        | stmt print exp ';'                        {printf("\nValue is
: %d\n",$3);}
        | stmt end ';'
        {exit(EXIT_SUCCESS);};

assign : identifier '=' exp                 {update($1,$3);}
;

exp     : term                {$$ = $1;}
        | '(' exp ')'  {$$ = $2;}
        | exp '=' exp  {$$ = $3;}
        | exp '+' exp  {$$ = $1+$3;}
        | exp '-' exp  {$$ = $1-$3;}
        | exp '*' exp  {$$ = $1*$3;}
        | exp '/' exp  {$$ = $1/$3;}
;

term : identifier     {$$ = value($1);}
        | num                 {$$ = $1;}
;

%%

int idx(char s)
{
        int i = -1;
        if(islower(s))
        {
            i = s - 'a' + 26;
        }
        else if(isupper(s))
        {
            i = s-'A';
        }
        return i;
}

int value(char s)
{
        int i = idx(s);
        return sym[i];
}

void update(char s,int val)
{
        int i = idx(s);
        sym[i] = val;
}

int main(void)
{
        int j;

        for(j = 0; j < 52; j++)
                sym[j] = 0;

        return yyparse();
}
```

```
void yyerror()
{

}
```

Output:

```
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ yacc -d bas3.y
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ lex bas3.l
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ gcc lex.yy.c y.tab.c -o bas3
bas3.l: In function ‘yylex’:
bas3.l:17:8: warning: implicit declaration of function ‘yyerror’; did you mean ‘
perror’? [-Wimplicit-function-declaration]
   17 | . {ECHO; yyerror(); }
      |        ^~~~~~~
      |        perror
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$ ./bas3
a=5;
print a+5
Value is 10
exit
master@master-VirtualBox:~/6th sem/cd_lab/lab 4$
```

Result:

Thus, the experiment has been successfully executed and verified.