Compiler Design Lab

Assignment-7
Generation of Intermediate Code using Lex and Yacc

Name : Vikraman S
Reg No. : 185001195

Code:

intermediate.l

```
%{
#include<stdio.h>
#include "y.tab.h"
extern YYSTYPE yylval;
%}

id [a-zA-Z_]+
no [0-9]+
ro ("<"|"<="|">"|">="|"=="|"!=")
nl "\n"
sp "\t"|" "

%%
"begin" {return BEG;}
"end" {return END;}
"if" {return IF;}
"then" {return THEN;}
"else" {return ELSE;}
"endif" {return ENDIF;}
{id} {yylval.var=strdup(yytext);return ID;}
{no} {yylval.val=atoi(yytext);return NUM;}
{ro} {return RELOP;}
{nl} {return NL;}
{sp} ;
. return yytext[0];
%%

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

intermediate.y

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

int yyerror(char *err);
int yylex(void);
int c=0;
char res[100],temp[50];
```

```
%}

%token ID NUM RELOP BEG END IF THEN ELSE ENDIF NL
%union{
    int val;
    char* var;
}

%type<val> NUM E F T
%type<var> ID

%left '<' '>'

%right '*' '/'

%left '+' '-'

%left '(' ')'

%%
G: BEG NL S NL END {sprintf(temp,"\n\nSyntactically
correct\n");strcat(res,temp);printf("%s",res);return 1;}
S: IF C THEN NL A NL ELSE NL A NL ENDIF
C: X RELOP X
X: ID | NUM
A: ID '=' E ';' {sprintf(temp,"\n%s = t%d",$1,$3);strcat(res,temp);}
E: E '*' T {sprintf(temp,"\nt%d = t%d %s
t%d",++c,$1,"*",$3);strcat(res,temp);$$=c;}
 | E '/' T {sprintf(temp,"\nt%d = t%d %s
t%d",++c,$1,"/",$3);strcat(res,temp);$$=c;}
 | E '%' T {sprintf(temp,"\nt%d = t%d %s
t%d",++c,$1,"%",$3);strcat(res,temp);$$=c;}
 | T {$$=$1;}
T: T '+' F {sprintf(temp,"\nt%d = t%d %s
t%d",++c,$1,"+",$3);strcat(res,temp);$$=c;}
 | T '-' F {sprintf(temp,"\nt%d = t%d %s t%d",++c,$1,"-
",$3);strcat(res,temp);$$=c;}
 | F {$$=$1;}
F: ID {sprintf(temp,"\nt%d = %s",++c,$1);strcat(res,temp);$$=c;}
 | NUM {sprintf(temp,"\nt%d = %d",++c,$1);strcat(res,temp);$$=c;}
%%

void main()
{
    yyparse();
    printf("\n");
}

int yyerror(char *err)
{
    printf(" Invalid - %s\n",err);
    exit(0);
}
```

Screenshot:

```
viki@viki:~/Desktop/CD Lab/Ex7/working$ lex intermediate.l
viki@viki:~/Desktop/CD Lab/Ex7/working$ yacc -d -Wnone intermediate.y
viki@viki:~/Desktop/CD Lab/Ex7/working$ gcc y.tab.c lex.yy.c
viki@viki:~/Desktop/CD Lab/Ex7/working$ cat input && ./a.out<input
begin
        if a<b then
                x=a*b+c/d;
        else
                x=a;
        endif
end

t1 = a
t2 = b
t3 = c
t4 = t2 + t3
t5 = t1 * t4
t6 = d
t7 = t5 / t6
x = t7
t8 = a
x = t8

Syntactically correct

viki@viki:~/Desktop/CD Lab/Ex7/working$ S
```