COURSE NO: CSE 3212

Project Name: Compiler design using flex and Bison

SUBMITTED BY

TUSHAR GHOSH JOY

ROLL:1307060

INPUT FILE :

```
shuru()
{

        int c shoman 0 ;
        int g shoman 9 ;
        int d shoman 12 ;

        d shoman c guun 10 plus g guun 11 plus 20 ;

        jodi (1>2)  c shoman c plus 12 guun 34 ;
        najodi  c shoman 10 ;


nirbachon ( g ) {

                khetro 10 :
                  c shoman c plus 5 ;
                khetro 6  :
                  c shoman c plus 7 ;
                khetro 9  :
                 c shoman c plus 10 ;
                thikvalue :
                  c shoman c plus 4 ;
        }

        int j shoman 10 ;

        ghurao (j>7)
        {

                j shoman j minus 1 ;

        }
}
```

FLEX FILE :

```
/* C Declarations */

%{
        #include<stdio.h>
        #include "main.tab.h"
        #include<stdlib.h>
        extern int yylval;
%}

/* RE and Actions */

%%

[0-9]+  {
                        yylval = atoi(yytext);
                        return NUM;
                }

[a-z]    {
                        yylval = *yytext - 'a';
                        return  VAR;
                }

"jodi"   {
                        return IF;
                }
"najodi"        {
                        return ELSE;
                }

[-+/*<>=,(){};:]        {
                                yylval = yytext[0];
                                return *yytext;
                        }



"vag"           { return '/'            ;}
"minus"                 { return '-'            ;}
"guun"          { return '*'            ;}
"plus"          { return '+'            ;}
"shoman"        { return '='            ;}

"shuru"     { return(VOIDMAIN)      ;}
```

```
"print"		{ return PRINT		;}

"int"	{ return(INT)		;}

"float"		{ return(FLOAT)		;}

"char"		{ return(CHAR)		;}

"ghurao"	{ return LOOP		;}
"khetro"		{ return CASE		;}

"thikvalue"	{return DEFAULT	;}

"nirbachon"	{ return SWITCH	;}

[ \t\n]* ;

.	{

		yyerror("Unknown Character.\n");
	}
%%

main(){
	yyin = freopen("in.txt","r",stdin);
	//yyout = freopen("out.txt","w",stdout);
	yyparse();
}
```

BISON FILE :

```
/* C Declarations */

%{
	#include<stdio.h>
	#include <math.h>
	#define YYSTYPE int
	int sym[26];
	int freq[26];
	int if_flag = 1, if_else_flag = 1, check = 1;
	int value ;
	int op1,op2,op3,operator ;
	int c1,c2,op;
	int f1 = 0 ;

	int casevalue[100] ;
	int casestatement[100];
	int sop1[100],sop2[100],sop3[100],soperator[100] ;
	int p = 0 ,s= 0;


	void kaj()
	{

			if(operator==1)
			 sym[op1] = sym[op2] +op3;

			 if (operator==2)
			 sym[op1] = sym[op2] -op3;


			 if (operator==3)
			 sym[op1] = sym[op2] *op3;

			 if (operator==4)
```

```
                {
                        sym[op1] = sym[op2] / op3;
                }
        printf("result is : %d\n",sym[op1]);
}


int skaj(int i)

{

                if(soperator[i]==1)
                sym[sop1[i]] = sym[sop2[i]] +sop3[i];

                if (soperator[i]==2)
                sym[sop1[i]] = sym[sop2[i]] -sop3[i];


                if (soperator[i]==3)
                sym[sop1[i]] = sym[sop2[i]] *sop3[i];

                if (soperator[i]==4)
                {
                        sym[sop1[i]] = sym[sop2[i]] / sop3[i];
                }
        return sym[sop1[i]] ;
}




%}

/* bison declarations */
```

```
%token NUM VAR IF ELSE VOIDMAIN INT FLOAT CHAR ID PRINT
LOOP  CASE DEFAULT SWITCH

%nonassoc IFX

%nonassoc ELSE

%left '<' '>'

%left '+' '-'

%left '*' '/'

/* Grammar rules and actions follow.  */

%%

program: VOIDMAIN '(' ')' '{' bstatement '}' //{printf("void main function");}
        ;




bstatement: /* empty */                 //{printf("start\n");}
         | bstatement statement         //{printf("b s \n");}
         ;



statement: ';'                          //{printf("sem\n");}

        | declartion ';'                //{printf("d sem\n");}

        | expression ';'                {
```

```
                                        //      printf("value of expression:
%d\n", $1);

                                    }
;

        |  SWITCH '(' expression ')'  '{'  kajkor  '}'     {

                                                value = $3 ;

                                                int v =0 ;
                                                int f2 = 1 ;



                                                for ( v=0;v<p;v++)
                                                {

                                                        if (value == casevalue[v] )
                                                        {

                                                                printf("result of
evaluation is :  %d\n",skaj(v) );

                                                                f2 =0 ;

                                                        }

                                                }

                                                if (f2==1) {

                                                        printf("default value is :
%d\n",skaj(p));
```

```
                                        }
                                    }
;


        | IF '(' expression ')' statement %prec IFX  {
                                        if($3)
                                        {
                                            //printf("\nonly if true and
value: %d",$3);
                                            printf("\nvalue of
expression in IF: %d\n",$5);
                                            //if_else_flag = 0;
                                            if_flag = 1;
                                            check = 1;
                                        }

                                        else
                                        {
                                            if(if_flag == 1)
                                            {
                                                printf("condition
value zero in IF block\n");
                                                if_flag = 0;
                                                if_else_flag = 0;
                                                check = 1;
                                            }
                                        }

                                    }


        | IF '(' expression ')' statement ELSE statement {
                                            if($3  )
```

```
                                                {
                                                        if_flag = 0;
                                                        if_else_flag = 0;
                                                        //printf("\nonly else if
true and value: %d",$3);
                                                        printf("\nvalue of
expression in IF: %d\n",$5);
                                                        check = 1;
                                                }
                                                else
                                                {
                                                        if(if_else_flag == 1)
                                                        {
                                                                check = 1;
                                                                if_flag = 0;
                                                                if_else_flag = 0;
                                                                //printf("\nonly
else else true");
                                                                printf("\nvalue
of expression in ELSE: %d\n",$7);
                                                        }
                                                }
                                        }

;

        | LOOP '('check')'  '{' dowork1 '}'

                                {

                                        if ( c2 ==-1 )
                                        {
                                                //printf("%d  \n",sym[c1]) ;
                                                while (sym[c1])
                                                {
```

```
                kaj() ;

        }

}


if (op==1)
{
        //printf("%d  %d\n",sym[c1],c2) ;
        while(sym[c1]<c2)
        {

                kaj() ;

        }

}


if (op==2)
{

        //printf(" %d  %d ",sym[c1],c2);

        while(sym[c1] > c2  )
        {

                kaj() ;

        }

}

if (op==3)
{
```

```
                                        //printf("%d  %d\n",sym[c1],c2) ;
                                        while(sym[c1] == c2)
                                        {

                                                kaj() ;

                                        }

                                }

                        }

        ;



        | '{' statement_list '}'   { $$ = $2; }
        ;


dowork1 : VAR '=' VAR '+' NUM ';'  {
                                        op1 = $1    ;
                                        op2 = $3    ;
                                        op3 = $5    ;
                                        operator = 1    ;

                                }


        | VAR '=' VAR '-' NUM ';' {

                                        op1 = $1    ;
                                        op2 = $3    ;
                                        op3 = $5    ;
```

```
                        operator = 2    ;

                    }


    | VAR '=' VAR '*' NUM  ';'{

                        op1 = $1    ;
                        op2 = $3    ;
                        op3 = $5    ;
                        operator = 3    ;
                    }



    | VAR '=' VAR '/' NUM ';'{

                        op1 = $1    ;
                        op2 = $3    ;
                        op3 = $5    ;
                        operator = 4    ;
                    }

;



dowork :    VAR '=' VAR '+' NUM ';'  {
                        sop1[s] = $1      ;
                        sop2[s] = $3      ;
                        sop3[s] = $5      ;
                        soperator[s] = 1    ;
                        s++ ;
                    }
```

```
        | VAR '=' VAR '-' NUM ';' {

                                sop1[s] = $1      ;
                                sop2[s] = $3      ;
                                sop3[s] = $5      ;
                                soperator[s] = 2    ;
                                s++ ;

                        }


        | VAR '=' VAR '*' NUM  ';'{

                                sop1[s] = $1      ;
                                sop2[s] = $3      ;
                                sop3[s] = $5      ;
                                soperator[s] = 3    ;
                                s++ ;
                        }


        | VAR '=' VAR '/' NUM ';'{

                                sop1[s] = $1      ;
                                sop2[s] = $3      ;
                                sop3[s] = $5      ;
                                soperator[s] = 4    ;
                                s++ ;
                        }

;



check :      VAR
{
```

```
        c1= $1;
        c2= -1;
        //printf("only var");
}



| VAR '<' NUM
{
        c1= $1;
        c2= $3;
        op= 1;
        //printf("var < num");
}



| VAR '>' NUM
{
        c1=$1;
        c2=$3;
        op=2;
        //printf("var > num");
}



| VAR '="=' NUM
{
        c1=$1;
        c2=$3;
        op=4;
        //printf("var == num");
}
```

```
        ;



declartion: TYPE ID1
          ;


TYPE : INT                                                  //{printf("int\n");}
    | FLOAT                                                 //{printf("flt\n");}
    | CHAR                                                  //{printf("char\n");}
    ;

ID1:  ID1 ',' expression {
                    if (freq[$3]==0) freq[$3]++;
                    else printf("declaration error ");
                      }

        | expression   {
                if (freq[$1]==0) freq[$1]++;
                else printf("declaration error ");

                  }
  ;


statement_list: statement_list statement              //{printf("inside if or else
up\n");}

                |statement
                            { $$ = $1;   // printf("inside if or else down\n");
                             }
                ;
```

```
expression:     NUM                         { $$ = $1;  }

        | VAR                               { $$ = sym[$1]; //printf("e:var %d \n",
$1);
                                    }

        | VAR '=' expression        {
                                    $$ = $3 ;
                                    sym[$1] = $3;
                                    //printf("Value of the variable: %d sem
\t\n",$3);
                                    }

        | expression '+' expression  { $$ = $1 + $3; }

        | expression '-' expression  { $$ = $1 - $3; }

        | expression '*' expression  { $$ = $1 * $3; }

        | expression '/' expression  {      if($3)
                                    {
                                        $$ = $1 / $3;
                                    }
                                    else
                                    {
                                        $$ = 0;
                                        printf("\ndivision by zero\t");
                                    }
                                }

        | expression '<' expression  { $$ = $1 < $3  ;}

        | expression '>' expression  { $$ = $1 > $3  ;}

        | '(' expression ')'        { $$ = $2   ;}
        ;
```

```
kajkor : kaj kajkor                    {
                                                                    }
        | DEFAULT ':' dowork {

                        //printf("in default\n");

        }
        ;



kaj : CASE NUM ':'  dowork  {
                                casevalue[p] = $2 ;

                                p++ ;
                        }
                ;

%%

int yywrap()
{
return 1;
}



yyerror(char *s){
        printf( "%s\n", s);
}
```

OUTPUT FILE :

value of expression in IF: 408

value of expression in ELSE: 10
result of evaluation is : 20
result is : 9
result is : 8
result is : 7