MVJ COLLEGE OF ENGINEERING

DEPARTMENT OF CSE

COMPILER DESIGN LABORATORY-MVJ20CSL66

Program 1 Tokenizer with LEX for declarations in C language.

```
%{
#include<stdio.h>
int d=0,c=0,id=0,s=0;
%}
%%
int|float|char|double {printf("%s is a keyword\n",yytext);d++;}
"," {printf("%s is a special symbol\n",yytext);c++;}
[a-zA-Z][a-zA-Z0-9]* {printf("%s is an identifier\n",yytext);id++;}
";" {printf("%s is a special symbol\n",yytext);s++;}
%%
void main()
{
printf("enter the statement\n");
yylex();
if(((d==1)&&(c==0)&&(id==1)&&(s==1))||((d==1)&&(id>1)&&(c==(id-1))&&(s==1)))
printf("valid declaration statement");
else
printf("Invalid declaration statement");
}
```

Program 2 Tokenizer with LEX for assignment statement.

```
%{
#include<stdio.h>
int d=0,s=0,asgn=0,dd=0,id=0;
%}
%%
int|float|char|double {printf("%s is a keyword\n",yytext);d++;}
[a-zA-Z][a-zA-Z0-9]* {printf("%s is a identifier\n",yytext);id++;}
"="  {printf("%s is an operator\n",yytext);asgn++;}
[0-9]+ {printf("%s is a digit\n",yytext);dd++;}
";" {printf("%s is a special character\n",yytext);s++;}
%%
void main()
{
printf("enter the assignment statement");
yylex();
if(((d==1)&&(id==1)&&(asgn==1)&&(dd==1)&&(s==1))||((s==1)&&(id==1)&&(asgn==1)
&&(dd==1)))
printf("valid assignment statement");
else
printf("invalid assignment statement");
}
```

Program 3:-Validation of for statement using Lex and Yacc

Lex Part

```
%{
# include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
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];
%%
```

Yacc part

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM FOR LE GE EQ NE OR AND
%right '='
%left OR AND
%left '>' '<' LE GE EQ NE
%left '+' '-'
%left '*' '/'
```

```
%right UMINUS
%left '!'

%%

S : ST {printf("Input accepted\n"); exit(0);}
;
ST: FOR '(' E1 ';' E2 ';' E3 ')' DEF
;
DEF:'{' BODY '}'
|;
BODY  :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
        | E '+' '+'
        | E '-' '-'
        | ID
        | NUM
```

```
        ;


E1: 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
       ;
E3: E'+"+'
|E '-"-'
;
%%
void main() {
   printf("Enter the expression:\n");
   yyparse();
}
yyerror()
{
printf("error");
}
```

Program 4:-Evaluation of arithmetic expression with LEX and YACC.

Lex part

```
%{
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ {yylval=atoi(yytext);return num;}
\t ;
\n return 0;
. return yytext[0];
%%
```

yacc part

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token num
%left '+' '-'
%left '*' '/'
%%
start: exp{printf("%d\n",$$);}
;
exp: exp'+'exp{$$=$1+$3;}
| exp'-'exp{$$=$1-$3;}
| exp'*'exp{$$=$1*$3;}
| exp'/'exp
  {
   if($3==0)
   {
    yyerror("error");
    exit(0);
    }
    else
    $$=$1/$3;
   }
 |'('exp')' {$$=$2;}
 |num{$$=$1;}
;
%%

void main()
{
printf("enter the expression in terms of integers:\n");
```

```
yyparse();
}
yyerror()
{
printf("error");
}
```

Program5 Symbol table creation from a list of declarations.

Input File:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
char c;
double d,e;
float f;
getch();
}
```

C program

```
#include<stdio.h>
#include<string.h>
char type[4][7]={"char","int","float","double"},*s=" ,;\n";
int size[4]={1,2,4,8};
void main()
{
 char *token,*token1,currentline[400];
 int i,j=3000;
 FILE *f1;
 f1=fopen("a.txt","r");
 printf("\nDatatype\tVariable\tSize\tAddress\n");
 while((fgets(currentline,1000,f1))!=NULL)
 {
 token=strtok(currentline,s);
printf("%c",token);
 for(i=0;i<4;i++)
 {

 if(strcmp(token,type[i])==0)
 {
 while((token1=strtok(NULL,s))!=NULL)
 {
 printf("\n%s\t\t%s\t\t%d\t%d",type[i],token1,size[i],j);
j=j+size[i];
 }
 }
 }
 }
 fclose(f1);
}
```

Program 6: Parser for if statement

Lex part

```
%{
#include"y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
if return IF;
then return THEN;
{digit}+ return NUM;
{alpha}({alpha}|{digit})* return ID;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
"||" return OR;
"&&" return AND;
[ \t\n]
. return yytext[0];
%%
```

Yacc Part

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM IF THEN LE GE EQ NE OR AND ELSE
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+"-'
```

```
%left '*'/'
%%
S:ST  {printf("Input accepted.\n");exit(0);};
ST:IF'(' E2 ')' THEN ST1';'
 ;
ST1:E
   ;
E :ID'='E
     |E'+'+'
     |E'-'-'
     |'+'+'E
     |'-'-'E
     |ID
     |NUM
     ;
E2:E2'<'E2
|E2'>'E2
     |E2 LE E2
     |E2 GE E2
     |E2 EQ E2
     |E2 NE E2
     |E2 OR E2
     |E2 AND E2
     |ID
     |NUM
     ;
%%
void main()
{
  printf("Enter the exp");
```

```
  yyparse();
}
yyerror()
{
printf("invalid");
}
```

7. Three address code generation for assignment statements with Array references

Lex part

```
%{
#include"y.tab.h"
%}
%%
[a-z] {yylval.sym= (char)(yytext[0]);return LETTER;}
. {return yytext[0];}
\n {return 0;}
%%
void yyerror(char *str)
{
printf("Invalid Character");
}
int main()
{
printf("Enter Expression x =>");
yyparse();
return(0);
}
```

YACC part

```
%{
#include"y.tab.h"
#include<stdio.h>
char p='A'-1;
char q='P';
%}
%union
{
```

```
char sym;
}
%token <sym> LETTER
%type <sym> S
%type <sym> E
%type <sym> array
%left '+"-'
%left '*"/'
%%
S : E {printf("x = %c\n",$$);}
 |array {printf("x=%c\n",$$);}
;
E : LETTER {}
|E '+' E {p++;printf("\n %c = %c + %c\n",p,$1,$3);$$=p;}
|E '-' E {p++;printf("\n %c = %c - %c\n",p,$1,$3);$$=p;}
|E '*' E {p++;printf("\n %c = %c * %c\n",p,$1,$3);$$=p;}
|E '/' E {p++;printf("\n %c = %c / %c\n",p,$1,$3);$$=p;}
|'(' E ')' {$$=p;}
;
array:LETTER '[' LETTER ']' {p++;printf("\n%c=%c*4\n",p,$3);printf("%c = %c +
&%c\n",q,p,$1);$$=q;}
%%
```

```
~$ ./a.out
Enter Expression x =>a[i]

A=i*4
P = A + &a
x=P
~$ ./a.out
Enter Expression x =>a*b+c

 A = a * b

 B = A + c
x = B
```

Program 8: Three address code generation from —while statement.

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
FILE *fptr1, *fptr2;
char str[50];
char alpha='A';
fptr1 = fopen("ff.txt", "r");
   if (fptr1 == NULL)
   {
      printf("Cannot open file");
      exit(0);
   }
 fptr2 = fopen("sample.txt", "w");
  if (fptr2 == NULL)
   {
        printf("Cannot open file");
      exit(0);
   }
while (fgets(str, 50, fptr1) != NULL)
{
              for(int i=0;str[i]!='\n';i++)
              {

                  if(str[i]=='w'&&str[i+1]=='h'&&str[i+2]=='i'&&str[i+3]=='l'&&str[i+4]=='e')
                    {
                          int l=i+5;
                          fputc(alpha,fptr2);
                          fputs(" ",fptr2);
```

```
                    fputs("if",fptr2);
                    while(str[l]!='\n')
                {
                        char ch=str[l];
                        fputc(ch,fptr2);
                        l++;
                }
                    fputs("then goto",fptr2);
                    fputs(" ",fptr2);
                    char a=alpha+1;
                    fputc(a,fptr2);
                    fputs("\n",fptr2);
                    break;
                }//if
            else
                {
                    fputc(alpha,fptr2);
                    fputs(" ",fptr2);
                    fputs(str,fptr2);
                    fputs("\n",fptr2);
                    break;

                }//else
            }//for
alpha++;
}//while
}//main
```

OUTPUT

Input file

ff.txt

```
i=0;
s=0;
while(i<10)
s=s+i;
```

sample.txt(OUTPUT file) OPEN AS gedit sample.txt

```
A i=0;
B s=0;
C if(i<10)then goto D
D s=s+i;
```

Program 9:-

Construction of flow graph from list of three address statements.

a.c

```
#include <stdio.h>
#include <stdlib.h> // For exit()
int main()
{
FILE *fptr1, *fptr2,*fptr3;
char str[50];
char label;
   fptr1 = fopen("s.txt", "r");
   if (fptr1 == NULL)
   {
      printf("Cannot open file");
      exit(0);
   }
      fptr2 = fopen("c.txt", "w");

      if (fptr2 == NULL)
   {
         printf("Cannot open file");
      exit(0);
   }

while (fgets(str, 50, fptr1) != NULL) {
                for(int i=0;str[i]!='\n';i++)
                {
```

```
                if(str[i]=='g'&&str[i+1]=='o'&&str[i+2]=='t'&&str[i+3]=='o')
                {
                    label=str[i+5];
                printf("Label value is %c",label);
                break;
                }
        }

}
fptr3=fopen("s.txt", "r");
while (fgets(str, 50, fptr3) != NULL) {
if(str[0]=='1')
{     fputs("B1\n",fptr2);
      fputs(str,fptr2);
}
if(str[0]=='2')
{

      fputs(str,fptr2);
}
if(str[0]=='3')
{
      fputs("B2\n",fptr2);
      fputs(str,fptr2);
}
if(str[0]=='4')
{

      fputs(str,fptr2);
}
```

}}

Input File

s.txt

1.i=0

2.p=0

3.i=i+1

4.if(i>10) then goto 3

OUTPUT

Label value is 3

Output Open c.txt as gedit c.txt

B1

1.i=0

2.p=0

B2

3.i=i+1

4.if(i>10) then goto 3

Program 10: Implement Constant Propagation

```
#include <stdio.h>
#include <stdlib.h> // For exit()

int main()
{
FILE *fptr1, *fptr2;
 int i;
char filename[100],line[100], c;
 printf("enter the assignment expression with semicolon at end");
 scanf("%s",line);
    fptr1 = fopen("b.txt", "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    fptr2 = fopen("c.txt", "w");
    if (fptr2 == NULL)
    {
          printf("Cannot open file %s \n", filename);
        exit(0);
    }

    // Read contents from file
    c = fgetc(fptr1);
    while (c != EOF)
    {
        if(c==line[0])
        {
        i=2;
        while (line[i]!=';')
       {
        fputc(line[i], fptr2);
        i++;
        c = fgetc(fptr1);
        }
      }
        else
        {
              fputc(c, fptr2);
            c = fgetc(fptr1);
     }
    }


    fclose(fptr1);
    fclose(fptr2);
    return 0;
```

```
}
```

Output

```
b.txt
i=a+10;
b=a+20;
c=b+a;

cc f.c
./a.out
Enter the assignment statement with semicolon at the end
 a=3;

c.txt (open it using gedit c.txt)
i=3+10;
b=3+20;
c=b+3;
```