**ASSIGNMENT NO.**

NAME: ROHIT ANIL GUND

CLASS: BE COMP-1 ROLL NO.: 402055

**PROGRAM:**

**.L file**

%{

#include "header1.hpp"

#include "y.tab.h"

extern int yyerror(char \*e);

%}

%%

if {return IF;}

[()] {return yytext[0];}

[a-zA-Z]+ {strcpy(yylval.attr.place,yytext);

yylval.attr.truelist=NULL;

yylval.attr.falselist=NULL;

return NAME;}

[ \t]+

\n return 0;

. return yytext[0];

%%

int yywrap()

{

return 1;

}

**.Y file**

%{

#include "header1.hpp"

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

extern int yylex();

extern int yyparse();

extern int yyerror(char \*e);

struct quad quad[10];

int quadarrayptr=0;

int i=1;

char t[10];

%}

%union

{

char op[10];

char dtype[10];

struct quad q1;

int pos;

struct ParseTreeNode attr;

struct IntListNode \*list;

int int1;

}

%token IF

%type <int1>M

%token <attr>NAME

%type <attr>exp

%type <list>S

%type <list>stmt

%left '+' '-'

%left '\*' '/'

%left '(' ')'

%%

S: IF exp M stmt {backpatch($2.truelist,$3);

$$=merge($2.falselist,$4);

display();

}

;

stmt: NAME'='exp { strcpy(quad[quadarrayptr].op,"=");

strcpy(quad[quadarrayptr].arg1,$3.place);

strcpy(quad[quadarrayptr].res,$1.place);

strcpy(quad[quadarrayptr].arg2,"-");

quadarrayptr++;i++;

}

;

exp : exp '+' exp{ add\_quad("+",$1.place,$3.place);

strcpy($$.place,t);}

|exp '-' exp{ add\_quad("-",$1.place,$3.place);

strcpy($$.place,t);}

|exp '\*' exp{ add\_quad("\*",$1.place,$3.place);

strcpy($$.place,t);}

|exp '/' exp{ add\_quad("/",$1.place,$3.place);

strcpy($$.place,t);}

| exp '<' exp {

$$.truelist = makelist(quadarrayptr);

$$.falselist=makelist(quadarrayptr+1);

strcpy(quad[quadarrayptr].op,"<");

strcpy(quad[quadarrayptr].arg1,$1.place);

strcpy(quad[quadarrayptr].arg2,$3.place);

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++;i++;

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++; }

| exp '>' exp {

$$.truelist = makelist(quadarrayptr);

$$.falselist=makelist(quadarrayptr+1);

strcpy(quad[quadarrayptr].op,">");

strcpy(quad[quadarrayptr].arg1,$1.place);

strcpy(quad[quadarrayptr].arg2,$3.place);

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++;i++;

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++; }

| exp '>''=' exp {

$$.truelist = makelist(quadarrayptr);

$$.falselist=makelist(quadarrayptr+1);

strcpy(quad[quadarrayptr].op, ">=");

strcpy(quad[quadarrayptr].arg1,$1.place);

strcpy(quad[quadarrayptr].arg2,$4.place);

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++;i++;

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++; }

| exp '<''=' exp {

$$.truelist = makelist(quadarrayptr);

$$.falselist=makelist(quadarrayptr+1);

strcpy(quad[quadarrayptr].op, "<=");

strcpy(quad[quadarrayptr].arg1,$1.place);

strcpy(quad[quadarrayptr].arg2,$4.place);

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++;i++;

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++; }

| exp '=''=' exp {

$$.truelist = makelist(quadarrayptr);

$$.falselist=makelist(quadarrayptr+1);

strcpy(quad[quadarrayptr].op, "==");

strcpy(quad[quadarrayptr].arg1,$1.place);

strcpy(quad[quadarrayptr].arg2,$4.place);

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++;i++;

strcpy(quad[quadarrayptr].res,"goto");

quadarrayptr++; }

|NAME{strcpy($$.place,$1.place);}

;

M : {$$ = quadarrayptr;}

;

%%

int main()

{

printf("Enter Input String:");

yyparse();

}

extern int yyerror(char \*e)

{

printf("Error::%s\n",e);

}

void newtemp()

{

char temp[10];

sprintf(temp,"%d",i++);

strcpy(t,"t");

strcat(t,temp);

}

void add\_quad(char op[10],char arg1[10],char arg2[10])

{

strcpy(quad[quadarrayptr].op,op);

strcpy(quad[quadarrayptr].arg1,arg1);

strcpy(quad[quadarrayptr].arg2,arg2);

newtemp();

strcpy(quad[quadarrayptr].res,t);

quadarrayptr++;

}

void display()

{

int j;

printf("ICG");

printf("\nSr.No\top\targ1\targ2\tres\n");

for(j=0;j<i;j++)

{

if(strcmp(quad[j].op,"")==0)

{

char buffer[10];

sprintf(buffer,"%d",i);

strcpy(quad[j].res,"goto(");

strcat(quad[j].res,buffer);

strcat(quad[j].res,")");

}

printf("\n%d\t%s\t%s\t%s\t%s\n",j,quad[j].op,quad[j].arg1,quad[j].arg2,quad[j].res);

}

}

void backpatch(struct IntListNode \*p,int i)

{

char buffer[10];

sprintf(buffer,"%d",i);

while(p!=NULL)

{

strcpy(quad[p->val].res,"goto(");

strcat(quad[p->val].res,buffer);

strcat(quad[p->val].res,")");

p=p->next;

}

}

struct IntListNode \*merge(struct IntListNode \*p1,struct IntListNode \*p2)

{

if(p1==NULL)

{

return p2;

}

else

{ while(p1->next!=NULL)

{

p1=p1->next;

}

p1->next=p2;

return p1;

}

}

struct IntListNode \*makelist(int i)

{

struct IntListNode \*temp;

temp=malloc(sizeof(struct IntListNode));

temp->val=i;

temp->next=NULL;

return temp;

}

**OUTPUT:**

