IT250 ACD Lab Assignment 8

Srinivasa R
211IT070

Note: '$' refers to end of file and refers to end of input for the parser

q1.l

```
%{
      #include "q1.tab.h"
      #include <string.h>
%}

%%
[ \t\n]
"while" return WHILE;
"do" return DO;
"print" return PRINT;
"or"  return OR;
"and" return AND;
"if"  return IF;
"else"   return ELSE;
[0-9]+     {
      strcpy(yylval.str, yytext);
      return NUM;
}
[A-Za-z]([A-Za-z]|[0-9])*  {
      strcpy(yylval.str, yytext);
      return ID;
}
"(" return OP;
")" return CP;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
[ \n]+     {}
. {return yytext[0];}
"----" return STMT;
```

```
"$\n" return END;
%%

int yywrap()
{
     return 0;
}
```

q1.y

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

     int countline = 1;
     int countvar = 0;
     char ir[2000];
     int stack[100];
     int ifstack[100];
     int top = 0;
     int iftop = 0;
%}

%union{
     char str[2000];
}

%token END
%token ID NUM WHILE LE GE EQ NE OR AND STMT OP CP DO PRINT IF ELSE
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/' '%'
%left '!'
%right UMINUS

%type <str> EXPRN
%type <str> EXPRNS
%type <str> WBCK
%type <str> CODE
```

```
%type <str> S
%type <str> BODY
%type <str> IFSTMNT
%type <str> IFBCK
%type <str> IFBDY
%type <str> WSTMNT
%type <str> NUM
%type <str> ID

%%

S : CODE END {
sprintf(ir, "%s", $1);
return 0;}
     ;

CODE: WBCK {
     sprintf($$, "%s", $1);
}
          | IFBCK {
     sprintf($$, "%s", $1);
}
          | EXPRNS ';' {
     sprintf($$, "%s", $1);
}
          | CODE CODE {
     sprintf($$, "%s\n%s", $1, $2);
}
          ;

WBCK: WSTMNT '{' BODY '}' {
     sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, $3,
stack[--top]);
     countline++;
}
     | WSTMNT ';' {
     sprintf($$, "%s %d\ngoto %d", $1, countline + 1, stack[--top]);
     countline++;
}
     | WSTMNT EXPRN ';' {
     sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, ir,
stack[--top]);
     sprintf(ir, "\0");
     countline++;
}
```

```
     | WSTMNT WBCK {
     sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, $2,
stack[--top]);
     countline++;
}
     | WSTMNT IFBCK {
     sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, $2,
stack[--top]);
     countline++;
}
     | WSTMNT '{' '}' {
     sprintf($$, "%s %d\ngoto %d", $1, countline + 1, stack[--top]);
}
     ;

WSTMNT: WHILE OP EXPRN CP {
     int irStartLine = countline - 1;
     for(int i = 0; i < strlen(ir); i++)
     {
     if(ir[i] == '\n')
     {
          irStartLine--;
     }
     }
     if(ir[0] == '\0') irStartLine = countline;
     sprintf($$, "%s\nif(%s == 0) goto ", ir, $3);
     sprintf(ir, "\0");
     if($$[0] == '\n')
     {
     for(int i = 0; i < strlen($$); i++)
     {
          $$[i] = $$[i + 1];
     }
     }
     stack[top] = irStartLine;
     top++;
     countline++;
}
               ;

EXPRNS: EXPRN {
     $$[0] = '\0';
     sprintf($$, "%s", ir);
     sprintf(ir, "\0");
}
```

```
      |PRINT EXPRN {
      sprintf($$, "print %s", $2);
      countline++;
}
      |EXPRNS ';' EXPRN {
      sprintf($$, "%s\n%s", $1, ir);
      sprintf(ir, "\0");
}
      |EXPRNS ';' PRINT EXPRN {
      sprintf($$, "%s\n%sprint %s", $1, ir, $4);
      sprintf(ir, "\0");
      countline++;
}
      ;

IFSTMNT: IF OP EXPRN CP {
      int irStartLine = countline - 1;
      for(int i = 0; i < strlen(ir); i++)
      {
      if(ir[i] == '\n')
      {
            irStartLine--;
      }
      }
      if(ir[0] == '\0') irStartLine = countline;
      sprintf($$, "%s\nif(%s == 0) goto ", ir, $3);
      sprintf(ir, "\0");
      if($$[0] == '\n')
      {
      for(int i = 0; i < strlen($$); i++)
      {
            $$[i] = $$[i + 1];
      }
      }
      countline++;
}
      ;

IFBCK: IFSTMNT IFBDY {
      sprintf($$, "%s %d\n%s %d", $1, countline + 1, $2, countline);
      top--;
}
      | IFSTMNT IFBDY ELSE IFBDY {
      int elseend = ifstack[--top];
      int ifend = ifstack[--top];
```

```
     sprintf($$, "%s %d\n%s %d\n%s %d", $1, ifend, $2, elseend, $4,
countline);
}
     ;

IFBDY: '{' BODY '}' {
     sprintf($$, "%s\ngoto", $2);
     ifstack[top] = ++countline;
     top++;
}
     | EXPRN ';' {
     sprintf($$, "%s\ngoto", ir);
     sprintf(ir, "\0");
     ifstack[top] = ++countline;
     top++;
}
     | IFBCK {
     sprintf($$, "%s\ngoto", $1);
     ifstack[top] = ++countline;
     top++;
}
     | WBCK {
     sprintf($$, "%s\ngoto", $1);
     ifstack[top] = ++countline;
     top++;
}
     | '{' '}' {
     sprintf($$, "goto");
     ifstack[top] = ++countline;
     top++;
}

BODY: WBCK {
     sprintf($$, "%s", $1);
}
     |EXPRNS ';' {
     sprintf($$, "%s", $1);
}
     |IFBCK {
     sprintf($$, "%s", $1);
}
     |BODY BODY {
     sprintf($$, "%s\n%s", $1, $2);
}
     ;
```

```
EXPRN: EXPRN '+' EXPRN {
      sprintf(ir, "%s\nt%d = %s + %s", ir, countvar, $1, $3);
      $$[0] = '\0';
      sprintf($$, "t%d", countvar);
      if(ir[0] == '\n')
      {
      for(int i = 0; i < strlen(ir); i++)
      {
            ir[i] = ir[i + 1];
      }
      }
      countvar++;
      countline++;
}
      |'-' EXPRN %prec UMINUS {
      sprintf(ir, "%s\nt%d = uminus %s", ir, countvar, $2);
      $$[0] = '\0';
      sprintf($$, "t%d", countvar);
      countvar++;
      if(ir[0] == '\n')
      {
      for(int i = 0; i < strlen(ir); i++)
      {
            ir[i] = ir[i+1];
      }
      }
      countline++;
}
      |EXPRN '*' EXPRN {
      sprintf(ir, "%s\nt%d = %s * %s", ir, countvar, $1, $3);
      $$[0] = '\0';
      sprintf($$, "t%d", countvar);
      if(ir[0] == '\n')
      {
      for(int i = 0; i < strlen(ir); i++)
      {
            ir[i] = ir[i + 1];
      }
      }
      countvar++;
      countline++;
}
      |EXPRN '-' EXPRN {
```

```
	sprintf(ir, "%s\nt%d = %s - %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	countvar++;
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countline++;
}
	|EXPRN '/' EXPRN {
	sprintf(ir, "%s\nt%d = %s / %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
	|EXPRN '%' EXPRN {
	sprintf(ir, "%s\nt%d = %s %% %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
	|EXPRN '<' EXPRN {
	sprintf(ir, "%s\nt%d = %s < %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
```

```
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
	|EXPRN '>' EXPRN {
	sprintf(ir, "%s\nt%d = %s > %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
	|EXPRN '=' EXPRN {
	sprintf(ir, "%s\n%s = %s", ir, $1, $3);
	$$[0] = '\0';
	sprintf($$, "%s", $1);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countline++;
}
	|EXPRN OR EXPRN {
	sprintf(ir, "%s\nt%d = %s or %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
```

```
            ir[i] = ir[i + 1];
        }
        }
        countvar++;
        countline++;
}
        |EXPRN AND EXPRN {
        sprintf(ir, "%s\nt%d = %s and %s", ir, countvar, $1, $3);
        $$[0] = '\0';
        sprintf($$, "t%d", countvar);
        if(ir[0] == '\n')
        {
        for(int i = 0; i < strlen(ir); i++)
        {
            ir[i] = ir[i + 1];
        }
        }
        countvar++;
        countline++;
}
        |'!' EXPRN {
        sprintf(ir, "%s\nt%d = !%s", ir, countvar, $2);
        $$[0] = '\0';
        sprintf($$, "t%d", countvar);
        if(ir[0] == '\n')
        {
        for(int i = 0; i < strlen(ir); i++)
        {
            ir[i] = ir[i + 1];
        }
        }
        countvar++;
        countline++;
}
        |EXPRN GE EXPRN {
        sprintf(ir, "%s\nt%d = %s >= %s", ir, countvar, $1, $3);
        $$[0] = '\0';
        sprintf($$, "t%d", countvar);
        if(ir[0] == '\n')
        {
        for(int i = 0; i < strlen(ir); i++)
        {
            ir[i] = ir[i + 1];
        }
        }
```

```
	countvar++;
	countline++;
}
	|EXPRN EQ EXPRN {
	sprintf(ir, "%s\nt%d = %s == %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
	|EXPRN NE EXPRN {
	sprintf(ir, "%s\nt%d = %s != %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
	|EXPRN LE EXPRN {
	sprintf(ir, "%s\nt%d = %s <= %s", ir, countvar, $1, $3);
	$$[0] = '\0';
	sprintf($$, "t%d", countvar);
	if(ir[0] == '\n')
	{
	for(int i = 0; i < strlen(ir); i++)
	{
		ir[i] = ir[i + 1];
	}
	}
	countvar++;
	countline++;
}
```

```
      |OP EXPRN CP {
      $$[0] = '\0';
      sprintf($$, "%s", $2);
}
      |NUM {
      sprintf($$, "%s", $1);
}
      |ID {
      sprintf($$, "%s", $1);
}
      ;

%%

int yyerror()
{
      printf("Parsing is failed.\n");
      return 0;
}

int main()
{
      ir[0] = '\0';
      stack[0] = 1;
      ifstack[0] = 1;
      yyparse();
      countline = 2;
      printf("1. ");
      for(int i = 0; i < strlen(ir); i++)
      {
      if(ir[i] == '\n')
      {
            printf("\n%d. ", countline);
            countline++;
      }
      else
            printf("%c", ir[i]);
      }
      printf("\n");
      return 0;
}
```

Outputs:

```
(base) wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 8$ ./q1
while(a<c or c>d)
{
        if(x<y and y<z or z<x)
                z = x + y*w;
        else
                z = z + 1;
}
$
1. t0 = a < c
2. t1 = c > d
3. t2 = t0 or t1
4. if(t2 == 0) goto  19
5. t3 = x < y
6. t4 = y < z
7. t5 = t3 and t4
8. t6 = z < x
9. t7 = t5 or t6
10. if(t7 == 0) goto  15
11. t8 = y * w
12. t9 = x + t8
13. z = t9
14. goto 18
15. t10 = z + 1
16. z = t10
17. goto 18
18. goto 1
```

```
(base) wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 8$ ./q1
while(A<C and B>D)
{
        if(A == 1)
                C = C + 1;
        else
                while(A <= D)
                        A = A + B;
}
$
1. t0 = A < C
2. t1 = B > D
3. t2 = t0 and t1
4. if(t2 == 0) goto  17
5. t3 = A == 1
6. if(t3 == 0) goto  10
7. t4 = C + 1
8. C = t4
9. goto 16
10. t5 = A <= D
11. if(t5 == 0) goto  15
12. t6 = A + B
13. A = t6
14. goto 10
15. goto 16
16. goto 1
```

```
(base) wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 8$ ./q1
a = 100;
while(a != 1)
{
        if(a % 2 == 0)
        {
                a = a / 2;
        }
        else
        {
                a = 3*a + 1;
        }
}
$
1. a = 100
2. t0 = a != 1
3. if(t0 == 0) goto  15
4. t1 = a % 2
5. t2 = t1 == 0
6. if(t2 == 0) goto  10
7. t3 = a / 2
8. a = t3
9. goto 14
10. t4 = 3 * a
11. t5 = t4 + 1
12. a = t5
13. goto 14
14. goto 2
```

```
(base) wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 8$ ./q1
a = 100;
count = 0;
while(a != 1)
{
        if(a%2 == 0)
                a = a / 2;
        else
                a = 3*a + 1;
        count = count + 1;
}
$
1. a = 100
2. count = 0
3. t0 = a != 1
4. if(t0 == 0) goto  18
5. t1 = a % 2
6. t2 = t1 == 0
7. if(t2 == 0) goto  11
8. t3 = a / 2
9. a = t3
10. goto 15
11. t4 = 3 * a
12. t5 = t4 + 1
13. a = t5
14. goto 15
15. t6 = count + 1
16. count = t6
17. goto 3
```