IT250 ACD Lab Assignment 7

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;
[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 top = 0;
%}

%union{
      char str[2000];
}

%token END
%token ID NUM WHILE LE GE EQ NE OR AND STMT OP CP DO PRINT
%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> WBDY
%type <str> WSTMNT
%type <str> NUM
%type <str> ID
```

```
%%

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

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

WBCK: WSTMNT '{' WBDY '}' {
     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 '{' '}' {
     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++;
}
      ;

WBDY: WBCK {
```

```
      sprintf($$, "%s", $1);
}
      |EXPRNS ';' {
            sprintf($$, "%s", $1);
      }
      |WBDY WBDY {
            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\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;
      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:

```
wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 7$ ./q1
a = 1;
b = 1;
while( a <= 5 )
{
b = 1;
while( b <= 5 )
{
b = b + 1;
print b;
}
a = a + 1;
print a;
}
$
1. a = 1
2. b = 1
3. t0 = a <= 5
4. if(t0 == 0) goto  16
5. b = 1
6. t1 = b <= 5
7. if(t1 == 0) goto  12
8. t2 = b + 1
9. b = t2
10. print b
11. goto 6
12. t3 = a + 1
13. a = t3
14. print a
15. goto 3
```

```
wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 7$ ./q1
while(a<c or c>d)
{
a=b/c*d+(-c);
print a;
}
$
1. t0 = a < c
2. t1 = c > d
3. t2 = t0 or t1
4. if(t2 == 0) goto  12
5. t3 = b / c
6. t4 = t3 * d
7. t5 = uminus c
8. t6 = t4 + t5
9. a = t6
10. print a
11. goto 1
```

```
wolfram@cuboid:~/School/IT250 ACD/Lab Assignment 7$ ./q1
a = 1;
c = a + 10*(a / 3);
while(c)
{
        while(a)
        {
                print a;
                a = a - 1;
        }
        while(!a)
                a = a + 1;
        c = c - 1;
}
$
1. a = 1
2. t0 = a / 3
3. t1 = 10 * t0
4. t2 = a + t1
5. c = t2
6. if(c == 0) goto  20
7. if(a == 0) goto  12
8. print a
9. t3 = a - 1
10. a = t3
11. goto 7
12. t4 = !a
13. if(t4 == 0) goto  17
14. t5 = a + 1
15. a = t5
16. goto 12
17. t6 = c - 1
18. c = t6
19. goto 6
```