IT250 – AUTOMATA & COMPILER DESIGN

ASSIGNMENT 7

Name: **Sachin Prasanna**
Roll No.: **211IT058**

Notation:

$ - This sign indicates that the input is complete and now generate the intermediate code (output) for the inputted code. Hence, after the $ sign, the output will be displayed as per the code logic.
The $ symbol acts the terminator for the code.

Note:

Since the written code generates intermediate code for the while loop, the code written works for both the given inputs and generates the required output.

Code Written:

LEX Code:

```
%{
    #include <stdio.h>
    #include "ques1.tab.h"
%}
```

```
%%

[ \t\n]

"or"    {
            return OR;
        }

"and"   {
            return AND;
        }

"while" {
            return WHILE;
        }

"print" {
            return PRINT;
        }

[0-9]+  {
            strcpy(yylval.str, yytext); return NUM;
        }

[A-Za-z_]+ {
            strcpy(yylval.str, yytext); return ID;
            }

"<="    {
            return LE;
        }

">="    {
            return GE;
        }

"=="    {
            return EQ;
        }

"!="    {
            return NE;
        }
```

```
[ \n]+     {}

.       {
            return yytext[0];
        }

"$\n"   {
            return END;
        }

%%

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

YACC Code:

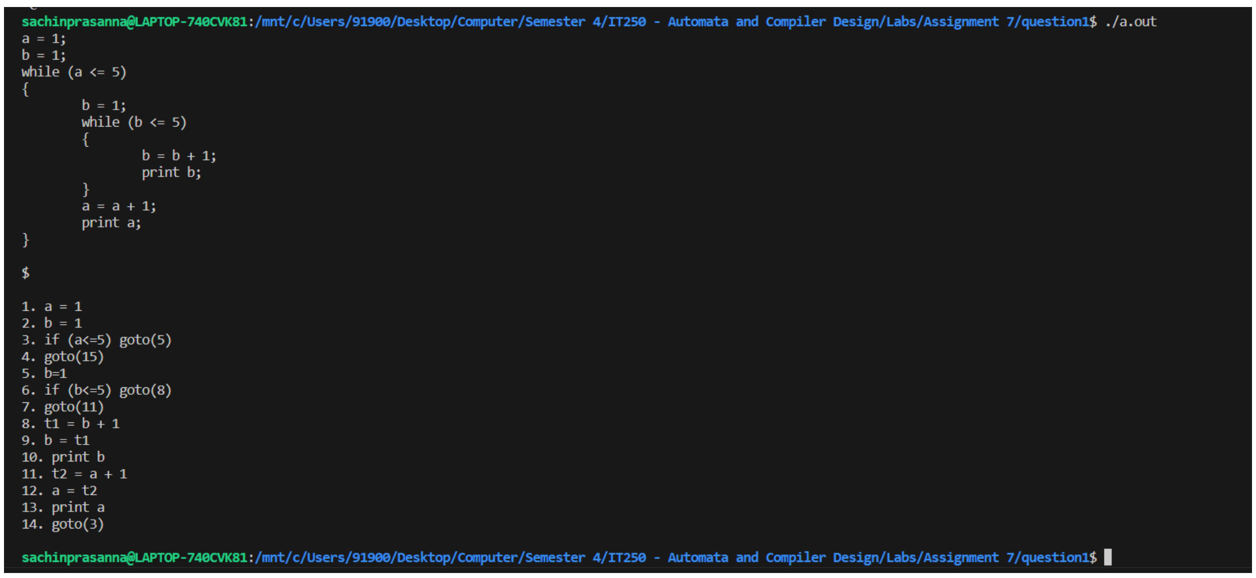
```
%{
    #include <stdio.h>
    #include <string.h>

    char outputBuffer[1000];
    int line = 1;
    int variable = 0, ptr = 0;
    int temp[1000];

    void update(){
        if(outputBuffer[0] == '\n'){
            for(int i = 0; i < strlen(outputBuffer); i++){
                outputBuffer[i] = outputBuffer[i + 1];
            }
        }
        variable++;
        line++;
    }

    void singleupdate(){
        if(outputBuffer[0] == '\n'){
            for(int i = 0; i < strlen(outputBuffer); i++){
                outputBuffer[i] = outputBuffer[i + 1];
            }
        }
        line++;
    }

    temp[0] = 1;
    outputBuffer[0] = '\0';

    void printIntermediateCode(){
        printf("1. ");
        line = 2;
        for(int i = 0; i < strlen(outputBuffer); i++) {
            if(outputBuffer[i] == '\n') printf("\n%d. ", line++);
            else printf("%c", outputBuffer[i]);
        }
        printf("\n");
    }

%}

%union{
    char str[1000];
}

%type <str> COND
%type <str> STMTS
%type <str> BLOCK
%type <str> STS
%type <str> S
%type <str> BODY
%type <str> STMT
%type <str> NUM
%type <str> ID

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

%%

S :         STS END { sprintf(outputBuffer, "%s", $1); return 0;}
```

```
            ;

STS:        BLOCK { sprintf($$, "%s", $1); }
            | STMTS ';' { sprintf($$, "%s", $1);}
            | STS STS { sprintf($$, "%s\n%s", $1, $2);}
            ;

BLOCK:      STMT '{' BODY '}' { sprintf($$, "%s %d\n%s\ngoto (%d)", $1, line + 1,
$3, temp[--ptr]); line++; }
            | STMT ';' { sprintf($$, "%s %d\ngoto (%d)", $1, line + 1, temp[--
ptr]); line++; }
            | STMT COND ';' { sprintf($$, "%s %d\n%s\ngoto (%d)", $1, line + 1,
outputBuffer, temp[--ptr]);
                              sprintf(outputBuffer, "\0");
                              line++; }
            | STMT BLOCK { sprintf($$, "%s %d\n%s\ngoto (%d)", $1, line + 1, $2,
temp[--ptr]); line++; }
            | STMT '{' '}' { sprintf($$, "%s %d\ngoto (%d)", $1, line + 1, temp[-
-ptr]); }
            ;

BODY:        BODY BODY { sprintf($$, "%s\n%s", $1, $2);}
            |BLOCK { sprintf($$, "%s", $1); }
            |STMTS ';' { sprintf($$, "%s", $1); }
            ;

STMT:       WHILE '(' COND ')' { sprintf($$, "%s\nif(%s == 0) goto ",
outputBuffer, $3);
                                 sprintf(outputBuffer, "\0"); temp[ptr] = line;
                                 ptr++; line++; }
            ;

STMTS:      COND { $$[0] = '\0'; sprintf($$, "%s", outputBuffer);
                   sprintf(outputBuffer, "\0"); }
            |PRINT COND { sprintf($$, "print %s", $2); line++; }
            |STMTS ';' COND { sprintf($$, "%s\n%s", $1, outputBuffer);
                              sprintf(outputBuffer, "\0");}
            |STMTS ';' PRINT COND { sprintf($$, "%s\n%sprint %s", $1,
outputBuffer, $4);
                                    sprintf(outputBuffer, "\0");
                                    line++; }
            ;

COND:       COND '+' COND { sprintf(outputBuffer, "%s\nt%d = %s + %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
```

```
                            sprintf($$, "t%d", variable);
                            update();}

            |COND '-' COND { sprintf(outputBuffer, "%s\nt%d = %s - %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND '*' COND { sprintf(outputBuffer, "%s\nt%d = %s * %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update();}

            |COND '/' COND { sprintf(outputBuffer, "%s\nt%d = %s / %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |'-' COND %prec UMINUS { sprintf(outputBuffer, "%s\nt%d = uminus %s",
outputBuffer, variable, $2); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            variable++;
                            singleupdate();}

            |COND OR COND { sprintf(outputBuffer, "%s\nt%d = %s or %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND AND COND { sprintf(outputBuffer, "%s\nt%d = %s && %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND LE COND { sprintf(outputBuffer, "%s\nt%d = %s <= %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND GE COND { sprintf(outputBuffer, "%s\nt%d = %s >= %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }
            |COND EQ COND { sprintf(outputBuffer, "%s\nt%d = %s == %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND NE COND { sprintf(outputBuffer, "%s\nt%d = %s != %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND '<' COND { sprintf(outputBuffer, "%s\nt%d = %s < %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND '>' COND { sprintf(outputBuffer, "%s\nt%d = %s > %s",
outputBuffer, variable, $1, $3); $$[0] = '\0';
                            sprintf($$, "t%d", variable);
                            update(); }

            |COND '=' COND { sprintf(outputBuffer, "%s\n%s = %s", outputBuffer,
$1, $3); $$[0] = '\0';
                            sprintf($$, "%s", $1);
                            singleupdate(); }

            |'(' COND ')' { $$[0] = '\0'; sprintf($$, "%s", $2); }

            |NUM { sprintf($$, "%s", $1); }

            |ID { sprintf($$, "%s", $1); }
            ;

%%

int main(){
    yyparse();
    printIntermediateCode();
    return 0;
}

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

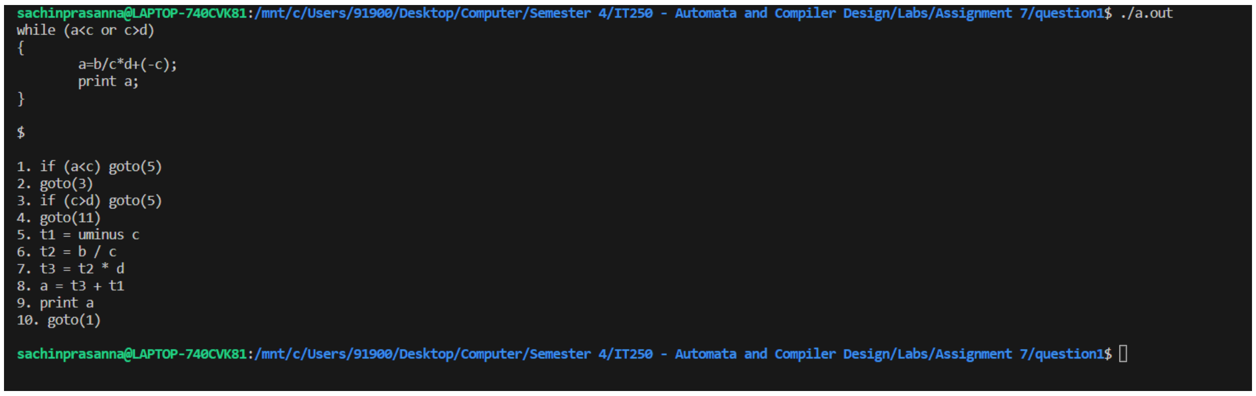
Outputs:

1)



```
sachinprasanna@LAPTOP-740CVK81:/mnt/c/Users/91900/Desktop/Computer/Semester 4/IT250 - Automata and Compiler Design/Labs/Assignment 7/question1$ ./a.out
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. if (a<=5) goto(5)
4. goto(15)
5. b=1
6. if (b<=5) goto(8)
7. goto(11)
8. t1 = b + 1
9. b = t1
10. print b
11. t2 = a + 1
12. a = t2
13. print a
14. goto(3)

sachinprasanna@LAPTOP-740CVK81:/mnt/c/Users/91900/Desktop/Computer/Semester 4/IT250 - Automata and Compiler Design/Labs/Assignment 7/question1$
```

2)



```
sachinprasanna@LAPTOP-740CVK81:/mnt/c/Users/91900/Desktop/Computer/Semester 4/IT250 - Automata and Compiler Design/Labs/Assignment 7/question1$ ./a.out
while (a<c or c>d)
{
        a=b/c*d+(-c);
        print a;
}

$

1. if (a<c) goto(5)
2. goto(3)
3. if (c>d) goto(5)
4. goto(11)
5. t1 = uminus c
6. t2 = b / c
7. t3 = t2 * d
8. a = t3 + t1
9. print a
10. goto(1)

sachinprasanna@LAPTOP-740CVK81:/mnt/c/Users/91900/Desktop/Computer/Semester 4/IT250 - Automata and Compiler Design/Labs/Assignment 7/question1$
```

****