Task -1

// input.txt

```c
int main()
{
    int num, original, reversed = 0, digit;
    printf("Enter an integer: ");
    scanf("%d", &num);
    original = num;
    do
    {
        digit = num % 10;
        reversed = reversed * 10 + digit;
        num = num / 10;
    } while (num != 0);
    printf("Reversed number: %d\n", reversed);
    if (original == reversed)
    {
        printf("The number is a palindrome.\n");
    }
    else
    {
        printf("The number is not a palindrome.\n");
    }
    return 0;
}
```

// cal.l

```lex
%option noyywrap

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

digit   [0-9]
number  {digit}+
id      [a-zA-Z_][a-zA-Z0-9_]*
ws      [ \t\n]+
string  \"([^\"\n]*)\"

%%

{ws}            {/* skip whitespace */}
{number}        { yylval.num = atoi(yytext); return NUM; }
{string}        { yylval.id = strdup(yytext); return STRING; }

"int"           { return INT; }
"main"          { return MAIN; }
"printf"        { return PRINTF; }
"scanf"         { return SCANF; }
"return"        { return RETURN; }
"if"            { return IF; }
"else"          { return ELSE; }
"do"            { return DO; }
"while"         { return WHILE; }
```

```
29
30    "=="                    { return EQ; }
31    "!="                    { return NEQ; }
32    "="                     { return ASSIGN; }
33    "%"                     { return MOD; }
34    "/"                     { return DIV; }
35    "*"                     { return MULT; }
36    "+"                     { return PLUS; }
37
38    "&"                     { return AMP; }
39    ";"                     { return SEMI; }
40    ","                     { return COMMA; }
41    "("                     { return LPAREN; }
42    ")"                     { return RPAREN; }
43    "{"                     { return LBRACE; }
44    "}"                     { return RBRACE; }
45
46    {id}                    { yylval.id = strdup(yytext); return ID; }
47
48    .                       { /* ignore unknown chars */ }
49
50    %%
51
```

//cal.y

```
1     %{
2     #include <stdio.h>
3     #include <stdlib.h>
4
5     int yylex();
6     void yyerror(const char *s);
7     %}
8
9     %union {
10        int num;
11        char* id;
12    }
13
14    %token <num> NUM
15    %token <id> ID
16    %token INT MAIN PRINTF SCANF RETURN IF ELSE DO WHILE
17    %token EQ NEQ ASSIGN MOD DIV MULT PLUS
18    %token LPAREN RPAREN LBRACE RBRACE SEMI COMMA
19    %token AMP STRING
20
21    %start program
22
23    %%
24
25    program:
26        INT MAIN LPAREN RPAREN LBRACE statements RBRACE
27        ;
28
29    statements:
30        statements statement
31        | statement
32        ;
33
```

```
34    statement:
35        declaration
36        | assignment
37        | print_stmt
38        | scan_stmt
39        | do_while_loop
40        | if_else
41        | return_stmt
42        ;
43
44    declaration:
45        INT id_list SEMI
46        ;
47
48    id_list:
49        ID
50        | ID ASSIGN expression
51        | id_list COMMA ID
52        | id_list COMMA ID ASSIGN expression
53        ;
54
55    assignment:
56        ID ASSIGN expression SEMI
57        ;
58
59    print_stmt:
60        PRINTF LPAREN STRING RPAREN SEMI
61        | PRINTF LPAREN STRING COMMA ID RPAREN SEMI
62        ;
63
64    scan_stmt:
65        SCANF LPAREN STRING COMMA AMP ID RPAREN SEMI
66        ;
67
```

```
68    expression:
69        NUM
70        | ID
71        | expression PLUS expression
72        | expression MULT expression
73        | expression DIV expression
74        | expression MOD expression
75        ;
76
77    do_while_loop:
78        DO LBRACE statements RBRACE WHILE LPAREN condition RPAREN SEMI
79        ;
80
81    condition:
82        expression EQ expression
83        | expression NEQ expression
84        ;
85
86    if_else:
87        IF LPAREN condition RPAREN LBRACE statements RBRACE ELSE LBRACE statements RBRACE
88        ;
89
90    return_stmt:
91        RETURN NUM SEMI
92        ;
93
94    %%
95
96    void yyerror(const char *s) {
97        fprintf(stderr, "Syntax error: %s\n", s);
98    }
99
100   int main() {
101       yyparse();
102       printf("Task-1 Parsing Done.\n");
103       return 0;
104   }
```

//output.txt

```
ACER@mdyasin MINGW64 /d/course-work/CSE_MU/CSE415-416_Compiler-Construction/Lab 09 - Test/SET B/task_1 (main)
$ make
bison -d cal.y
cal.y: conflicts: 16 shift/reduce
flex cal.l
gcc cal.tab.c lex.yy.c
./a.exe <input.txt> output.txt
```

Task -2

// input.txt

```c
int main()
{
    char op;
    double a, b, result;
    printf("Enter operation (+, -, *, /, ^): ");
    scanf(" %c", &op);
    printf("Enter two numbers: ");
    scanf("%lf %lf", &a, &b);
    switch (op)
    {
    case '+':
        result = a + b;
        printf("Result: %.2lf\n", result);
        break;
    case '-':
        result = a - b;
        printf("Result: %.2lf\n", result);
        break;
    case '*':
        result = a * b;
        printf("Result: %.2lf\n", result);
        break;
    case '/':
        if (b != 0)
            printf("Result: %.2lf\n", a / b);
        else
            printf("Error: Division by zero!\n");
        break;
    case '^':
        result = pow(a, b);
        printf("Result: %.2lf\n", result);
        break;
    default:
        printf("Invalid operator!\n");
    }
    return 0;
}
```

// cal.l

```lex
1    %{
2    #include "cal.tab.h"
3    #include <stdlib.h>
4    #include <string.h>
5    %}
6
7    %%
8
9    "int"           { return INT; }
10   "char"          { return CHAR; }
11   "double"        { return DOUBLE; }
12   "main"          { return MAIN; }
13   "return"        { return RETURN; }
14   "scanf"         { return SCANF; }
15   "printf"        { return PRINTF; }
16   "switch"        { return SWITCH; }
17   "case"          { return CASE; }
18   "default"       { return DEFAULT; }
19   "break"         { return BREAK; }
20   "if"            { return IF; }
21   "else"          { return ELSE; }
22
23   "!="            { return NEQ; }
24   "^"             { return POWER; }
25   "&"             { return AMP; }
26
27   ":"             { return COLON; }
28   ";"             { return SEMICOLON; }
29   "="             { return ASSIGN; }
30   ","             { return COMMA; }
31   "("             { return LPAREN; }
32   ")"             { return RPAREN; }
33   "{"             { return LBRACE; }
34   "}"             { return RBRACE; }
35
```

```lex
35
36   [0-9]+\.[0-9]+  { yylval.dval = atof(yytext); return NUMBER; }
37   [0-9]+          { yylval.dval = atof(yytext); return NUMBER; }
38   "\'."           { yylval.cval = yytext[1]; return CHAR_CONST; }
39
40   [a-zA-Z_][a-zA-Z0-9_]* { yylval.id = strdup(yytext); return ID; }
41
42   \".*\"          { return STRING; }
43
44   "+"             { return PLUS; }
45   "-"             { return MINUS; }
46   "*"             { return MUL; }
47   "/"             { return DIV; }
48
49   [ \t\r\n]+      { /* skip whitespace */ }
50
51   .               { return yytext[0]; }
52
53   %%
54
55   int yywrap() {
56       return 1;
57   }
58
```

// cal.y

```
 1   %{
 2   #include <stdio.h>
 3   #include <stdlib.h>
 4   #include <math.h>
 5
 6   double a, b, result;
 7   char op;
 8
 9   int yylex(void);
10   int yyerror(char *s);
11   %}
12
13   %union {
14       int ival;
15       char cval;
16       char *id;
17       double dval;
18   }
19
20   %token <ival> INT CHAR DOUBLE RETURN MAIN
21   %token <id> ID
22   %token <cval> CHAR_CONST
23   %token <dval> NUMBER
24   %token <id> STRING
25
26   %token IF ELSE SCANF PRINTF
27   %token SWITCH CASE DEFAULT BREAK
28   %token ASSIGN PLUS MINUS MUL DIV POWER
29   %token NEQ AMP
30   %token LPAREN RPAREN LBRACE RBRACE SEMICOLON COLON COMMA
31
32   %start program
33
34   %%
35
36   program:
37         INT MAIN LPAREN RPAREN LBRACE declarations statements RBRACE
38       ;
39
40   declarations:
41         declarations declaration
42       | declaration
43       ;
44
45   declaration:
```

```
39
40   declarations:
41         declarations declaration
42       | declaration
43       ;
44
45   declaration:
46         CHAR ID SEMICOLON
47       | DOUBLE id_list SEMICOLON
48       ;
49
50   id_list:
51         ID
52       | id_list COMMA ID
53       ;
54
55   statements:
56         statements statement
57       | statement
58       ;
59
60   statement:
61         scan_stmt
62       | print_stmt
63       | assign_stmt
64       | switch_stmt
65       | return_stmt
66       | if_stmt
67       ;
68
69   scan_stmt:
70         SCANF LPAREN STRING COMMA AMP ID RPAREN SEMICOLON
71       ;
72
73   print_stmt:
74         PRINTF LPAREN STRING RPAREN SEMICOLON
75       | PRINTF LPAREN STRING COMMA ID RPAREN SEMICOLON
76       ;
77
78   assign_stmt:
79         ID ASSIGN expr SEMICOLON
80       ;
81
```

```
 75    print_stmt:

 77
 78    assign_stmt:
 79          | ID ASSIGN expr SEMICOLON
 80          ;
 81
 82    switch_stmt:
 83          | SWITCH LPAREN ID RPAREN LBRACE case_list default_case RBRACE
 84          ;
 85
 86    case_list:
 87          | case_list case_stmt
 88          | case_stmt
 89          ;
 90
 91    case_stmt:
 92          | CASE CHAR_CONST COLON statements BREAK SEMICOLON
 93          ;
 94
 95    default_case:
 96          | DEFAULT COLON statements
 97          ;
 98
 99    return_stmt:
100          | RETURN NUMBER SEMICOLON
101          ;
102
103    if_stmt:
104          | IF LPAREN expr NEQ NUMBER RPAREN statement
105          | IF LPAREN expr NEQ NUMBER RPAREN statement ELSE statement
106          ;
107
```

```
108    expr:
109          | expr PLUS expr
110          | expr MINUS expr
111          | expr MUL expr
112          | expr DIV expr
113          | POWER LPAREN expr COMMA expr RPAREN
114          | ID
115          | NUMBER
116          ;
117
118    %%
119
120    int main() {
121          yyparse();
122          printf("Task-2 Parsing Done.\n");
123          return 0;
124    }
125
126    int yyerror(char *s) {
127          fprintf(stderr, "Error: %s\n", s);
128          return 0;
129    }
130
```

// output

```
SET B > task_2 > output.txt
  1    Task-2 Parsing Done.
  2
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

$ make
bison -d cal.y
cal.y: conflicts: 17 shift/reduce
flex cal.l
gcc cal.tab.c lex.yy.c -o a -lm
./a < input.txt > output.txt
Error: syntax error
```