NAME: Jinyi Xia
STUDENT ID: 2021212057
CLASS NUMBER: 2021211802
CONTAINER NUMBER: a4d4f6a5498a4e912a4d1a76a7fa36cf23af64b416688d222a8968a38b29a711

REPORT ON LAB 4

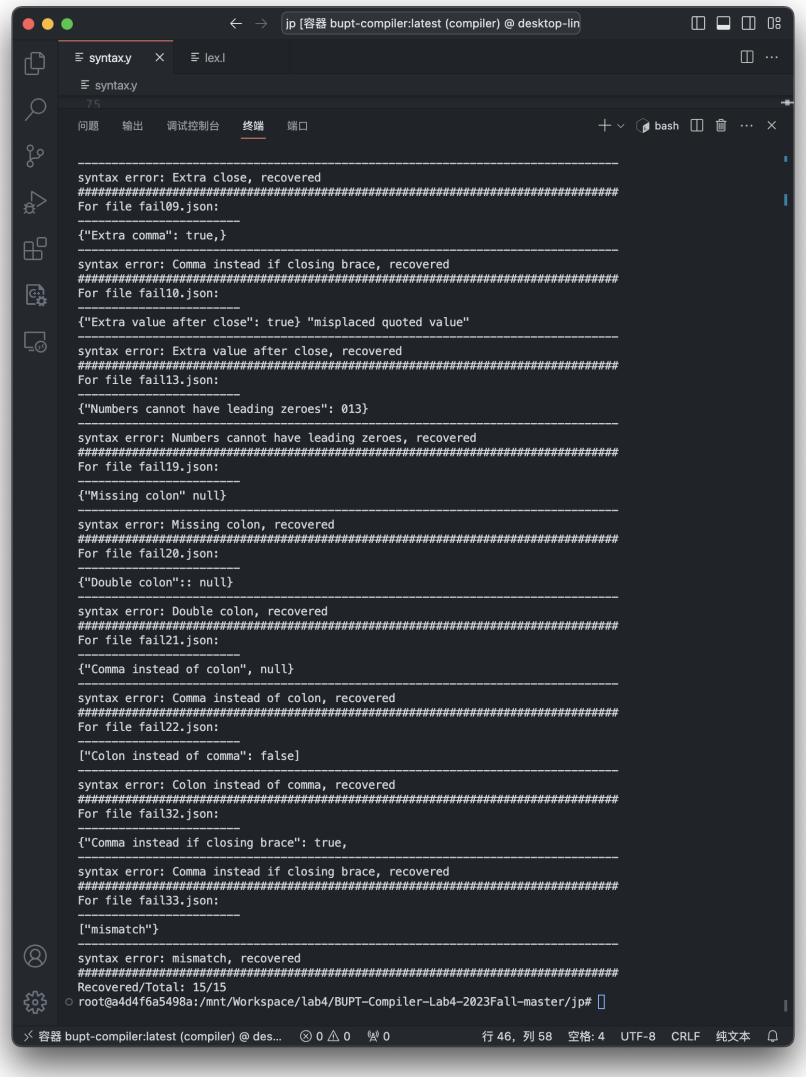

Figure 1: Test result

The implementation of `lex.l` is as follows.

```
%{
    #include"syntax.tab.h"
%}
%option noyywrap
```

```
5
6  unic u[0-9a-fA-F]{4}
7  esc \\([\"\\/bfnrt]|{unic})
8  scp [^"\\\x00-\x1f]
9  string \"({esc}|{scp})*\"
10
11 int 0|[1-9][0-9]*
12 frac \.[0-9]+
13 exp [Ee][+-]?[0-9]+
14 number -?{int}{frac}?{exp}?
15 leadingzero 0{int}
16
17 empty [ \n\r\t]
18
19 %%
20
21 "{" { return LC; }
22 "}" { return RC; }
23 "[" { return LB; }
24 "]" { return RB; }
25 ":" { return COLON; }
26 "," { return COMMA; }
27
28 "true" { return TRUE; }
29 "false" { return FALSE; }
30 "null" { return VNULL; }
31
32 {string} { return STRING; }
33 {number} { return NUMBER; }
34 {leadingzero} { return LEADINGZERO; }
35
36 {empty} {}
37
38 . { printf("lexical error: %s\n", yytext); }
```

The implementation of `syntax.y` is as follows.

```
1  %{
2      #include"lex.yy.c"
3      void yyerror(const char*);
4  %}
5
6  %token LC RC LB RB COLON COMMA
7  %token STRING NUMBER LEADINGZERO
8  %token TRUE FALSE VNULL
9  %%
10
11 Json:
12         Value
13     | Json COMMA error { puts("Comma after the close, recovered"); }
14     | Json RB error { puts("Extra close, recovered"); }
```

```
15      ;
16  Value:
17          Object
18      | Array
19      | STRING
20      | NUMBER
21      | TRUE
22      | FALSE
23      | VNULL
24      ;
25  Object:
26          LC RC
27      | LC Members RC
28      | Object Value error { puts("Extra value after close, recovered"); }
29      | LC Values RC error { puts("Comma instead of colon, recovered"); }
30      | LC Member COMMA error { puts("Comma instead if closing brace, recovered"); }
31      ;
32  Members:
33          Member
34      | Member COMMA Members
35      | Members COMMA error { puts("Extra comma, recovered"); }
36      ;
37  Member:
38          STRING COLON Value
39      | STRING COLON COLON Value error { puts("Double colon, recovered"); }
40      | STRING COLON LEADINGZERO error { puts("Numbers cannot have leading zeroes,
            recovered"); }
41      | STRING Value error { puts("Missing colon, recovered"); }
42      ;
43  Array:
44          LB RB
45      | LB Values RB
46      | LB Values RC error { puts("mismatch, recovered"); }
47      | LB Members RB error { puts("Colon instead of comma, recovered"); }
48      | LB Values error { puts("Unclosed array, recovered"); }
49      ;
50  Values:
51          Value
52      | Value COMMA error { puts("extra comma, recovered"); }
53      | Value COMMA COMMA error { puts("double extra comma, recovered"); }
54      | Value COMMA Values
55      | COMMA Values error { puts("missing value, recovered"); }
56      ;
57  %%
58
59  void yyerror(const char *s){
60      printf("syntax error: ");
61  }
62
63  int main(int argc, char **argv){
```

```
64      if(argc != 2) {
65          fprintf(stderr, "Usage: %s <file_path>\n", argv[0]);
66          exit(-1);
67      }
68      else if(!(yyin = fopen(argv[1], "r"))) {
69          perror(argv[1]);
70          exit(-1);
71      }
72      yyparse();
73      return 0;
74  }
```