Project 1 Report

Basic Requirements

1. Basic parser
2. Report lexical error and syntax error to pass some test cases

Basic parser

1. Using flex to recognize some pattern such as `int, float, char`. And for many type A error pattern, such as `'\xt0'(invalid char), 0x5gg(invalid integer), 0a(invalid identifier)`, our flex program are required to be able to match it and report correct error. For instance,

   ```
   struct abc {
       int a,1b
       float c,3d = 10.0
       char _e,$ = '\xt0';
   };
   ```

   Output:

   ```
   Error type A at Line 2: Unknown lexeme 1b
   Error type B at Line 3: Missing semicolon ';'
   Error type A at Line 3: Unknown lexeme 3d
   Error type B at Line 4: Missing semicolon ';'
   Error type A at Line 4: Unknown lexeme $
   Error type A at Line 4: Unknown lexeme '\xt0'
   ```

2. Using bison to analyze syntax, in the basic productions, we should implement some error recovery to pass the test cases, such as last example's type B error. When the program miss some tokens, we should report that and continue parsing. Be caution, if there are two simple error in the same line, we should report them both instead of only the first one.
3. For this parser, I only implement some basic syntax error recovery to pass the test cases, because there are too many types of error, and I think it is impossible to recovery all of them.

Test cases

For my own test cases

1. ```
   int test_1_r01(int a, int b) {c = 'c';if (a > b){return a;}else{return b;}}
   ```

   Start with a no error program in one line.

2. ```
   int func_1(int a){ return 3 * a;}int func_2(int x, int y){ return x + y;}int test_1_r04({  int a, b, c;  c = func_2(func_1(func_1(a+b*func_1(b)), 1.7); return c;}
   ```

   Then a missing two parentheses program in one line.

3.
```
struct abc {
    int a,b;
    float c,d;
    char e,f;
};
```

A structure related no error test case was provided.

4.
```
struct abc {
    int a,1b
    float c,3d = 10.0
    char _e,$ = '\xt0';
};
```

A structure related, with 2 syntax error and 3 lexical error, test case.

5.
```
struct abc {int a,1b float c,3d = 10.0 char _e,$ = '\x0';};
```

Previous test case in one line.

Compile and Test

This program can using `./bin/splc ./test/test_*.spl ./test_ex/test_*.spl` to evoke and the program will write output file with same name but different suffix in `.out`

The test cases I submitted are all passed.