12011619 Liquan Wang
12111744 Wenhui Tao
12111611 Ruixiang Jiang

1 cs323-project-phase1

Completion status：
Pass all basic sample tests， and add following features:

Please note:
Because of aesthetic issues, the code is partially commented, but the actual program is not commented
To make the code more intuitive, we've attached test samples and results that you can skip if you don't want to read them

- single- and multi-line comment
- macro preprocessor
- file inclusion
- for statements

1.1 single- and multi-line comment

Listing 1: Implementation

```
  /* in lex.l
  Recognize but do not react to ignore comments*/
  "//".* {}
  "/*"((("*"[^/])?)|[^*])*"*/"   {}
```

Listing 2: test

```
// this is a single Line
/* here is a A multi-line comment */
/* here is a /* A illegal*/ multi-line comment */
```

Listing 3: result

```
  Program (1)
```

1.2 macro preprocessor and file inclusion

Listing 4: lex.l Implementation

```
  /* in lex.l
  Enter the <macro> state after a special beginning
     is recognized to distinguish it from the
     exception recognition capture
  */
  "#include" {yylval=strdup("INCLUDE\n"); BEGIN(
     macro); return INCLUDE;}
  "#define" {yylval=strdup("DEFINE\n"); BEGIN(macro)
     ; return DEFINE;}
  "#ifdef" {yylval=strdup("IFDEF\n"); BEGIN(macro);
     return IFDEF;}
  "#else" {yylval=strdup("MACROELSE\n"); return
     MACROELSE;}
  "#endif" {yylval=strdup("ENDIF\n"); return ENDIF;}
  <macro>{
    "," {yylval=strdup("COMMA\n");return COMMA;}
    "<" {return LT;}
    ">" {return GT;}
//  \"  {return DQUOT;}
    \n  {BEGIN(INITIAL);}
    [ \t]+ /*ignore word splits*/{}
//  [^," "<>"\n]+ {asprintf(&yylval,"%s\n",yytext);
   return MACRO;}
}
```

Listing 5: syntax.y Implementation

```
  /* in syntax.y
  Macro, file introduction automatically changes to
     the beginning of the file, and parallel output,
      to achieve #define #ifdef #else #endif #
     include
  */
  /*You can define multiple macros on a single line
     separated by commas*/

  RES:
```

```
  Program
| MACROStmt Program

/* #define && #include*/
MACROStmt:
  INCLUDEStmt
| DEFINEStmt
| DEFINEStmt MACROStmt
| INCLUDEStmt MACROStmt

INCLUDEStmt:
  INCLUDE LT MACRO GT
| INCLUDE DQUOT MACRO DQUOT

DEFINEStmt:
  DEFINE MACRO MACRO
| DEFINEStmt COMMA MACRO MACRO

/*#ifdef #else #endif*/
Stmt:
....
| IFDEF Stmt ENDIF
| IFDEF MACRO Stmt MACROELSE Stmt ENDIF
```

Listing 6: test

```
#define MACRO_NAME replacement_text, MACRO_NAME2
   replacement_text2
#define MACRO_NAME1 replacement_text1
#include <header_file.h>
#include "header_file.h"
#define DEBUG 1

int main() {
#ifdef DEBUG
    printf('y');
#else
    printf('n');
#endif

```

```
    return 0;
}
```

Listing 7: result

```
DEFINE (1)
  MACRO_NAME
  replacement_text
DEFINE (1)
  MACRO_NAME2
  replacement_text2
DEFINE (2)
  MACRO_NAME1
  replacement_text1
INCLUDE (3)
  header_file.h
INCLUDE (4)
  header_file.h
DEFINE (5)
  DEBUG
  1
Program (7)
  ...
          Stmt (8)
            IFDEF
            DEBUG
            Stmt (9)
              Exp (9)
                ID: printf
                LP
                Args (9)
                  Exp (9)
                    CHAR: 'y'
                RP
              SEMI
            MACROELSE
            Stmt (11)
              Exp (11)
                ID: printf
                LP
                Args (11)
                  Exp (11)
                    CHAR: 'n'
                RP
              SEMI
    ...
```

1.3 for statements

Listing 8: lex.l Implementation

```
  for {yylval=strdup("FOR\n"); return FOR;}
```

Listing 9: syntax.y Implementation

```
/*The logic of for is very similar to "if" "while"*/
Stmt:
  ...
  | FOR LP Exp SEMI Exp SEMI Exp RP Stmt {asprintf(&
     $$,"Stmt␣(%d)\n%s\n", @$.first_line,
     concat_shift($1,$2,$3,$4,$5,$6,$7,$8,$9));}
```

Listing 10: FOR test

```
  int test_2(int a, int b)
{
   for (i = 0; i < 5; i=i+1) {
        printf('now␣it␣is␣%d␣\n', i);
    }
    return 0;
}
```

Listing 11: FOR test result

```
  Program (1)
      ...
            FOR
            LP
            Exp (3)
              Exp (3)
                ID: i
```

```
          ASSIGN
          Exp (3)
            INT: 0
        SEMI
        Exp (3)
          Exp (3)
            ID: i
          LT
          Exp (3)
            INT: 5
        SEMI
        Exp (3)
          Exp (3)
            ID: i
          ASSIGN
          Exp (3)
            Exp (3)
              ID: i
            PLUS
            Exp (3)
              INT: 1
        RP
        Stmt (3)
          ...
```