OurC Project 2

Proj. 2 is the first part of OurC project (there are three parts). For Proj. 2, you are to implement a syntax checker and a pretty printer that supports system-supported functions.

The system-supported functions of OurC system are listed below.

```
ListAllVariables();          // just the names of the (global) variables,
                             // sorted (from smallest to greatest)
ListAllFunctions();          // just the names of the (user-defined)
                             // functions, sorted
ListVariable(char name[]);   // the definition of a particular variable
ListFunction(char name[]);   // the definition of a particular function
Done();                      // exit the interpreter
```

在 error message 的部分，共有三種 error messages。其例如下：

```
Line 3 : unrecognized token with first char : '$'        // lexical error
Line 2 : unexpected token : '*'                          // syntactical error (token recognized)
Line 5 : undefined identifier : 'bcd'                    // semantic error (grammar ok)
```

說明(務必請遵守以下的 **error detection** 順序)：

在 get token 時，我們先找到 the first non-white-space char。

如果 the first non-white-space char 並不是任何 token 的 first char，那就是 lexical level (scanner level)的 error，印出 unrecognized-token-message。

如果這個 char 是一個或多個 token 的 first char、我們就 go for the longest match (and get that token)。此時若有文法上(syntactical level)的 error，就印出 unexpected-token-message。

如果沒有文法上的 error、而且 token 是個 identifier，就檢查 undeclared identifier (if this identifier should have been declared at this point)，如果是個 undeclared identifier (a semantic error), 就印 undefined-identifier-message。

There is a total of 14 "test problems" for Proj. 2.

Each test problem contains several tests. The first few tests are "viewable", that is, you will be able to see what the test datas are. The latter tests, however, are such that you will not see what the test data is.

Odd-numbered test problems are designed in such a way that the latter tests are more or less "isomorphic" to the "structures" of the first few tests. Even-numberd test problems are such that their latter tests are slightly more involved versions of their first few tests.

Though you can do the problems in any order, it is suggested that you do the problems by sequence. That is, you do Problem No. 1, followed by Problem No. 2, followed by Problem No. 3, etc. Another alternative is to do the

odd-numbered problems first. Then, proceed to solve the even-numbered problems. As a general guideline, you should always try the lower-numbered problems before you try the higher-numbered problems.

Below, we use example I/O to give you a "feel" of what your Proj. 2 should do.

// ==========

當輸入是>>2

Done();
<<

你的輸出應該是>>Our-C running ...

> Our-C exited ...<<

// ==========

// 不用擔心你的輸出的最後會有"trailing white-spaces"，

// 有 trailing white-spaces 也等於沒有，系統"看不到"你的輸出中最後的的 trailing white-spaces

// ==========

當輸入是>>2

int x ;
x=10;
cout << x ;
Done();
<<

你的輸出應該是>>Our-C running ...

> Definition of x entered ...
> Statement executed ...
> Statement executed ...
> Our-C exited ...<<

// ==========

當輸入是>>3

string str ;
str = "This is a fine day.\n" ;
float y ;
y = 20 ;
Done();
<<

你的輸出應該是>>Our-C running ...

```
> Definition of str entered ...
> Statement executed ...
> Definition of y entered ...
> Statement executed ...
> Our-C exited ...<<
```

// ==

當輸入是>>2

```
string str ;
str = "This is a fine day.\n" ;
str = str + "Isn't it?\n" ;
cout << str ;
float y ;
int x ;
y = 20 * ( x - y ) / 34.5 ;
cout << "Value of 'y' is now : " << y << "\n" ;
Done();
<<
```

你的輸出應該是>>Our-C running ...

```
> Definition of str entered ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Definition of y entered ...
> Definition of x entered ...
> Statement executed ...
> Statement executed ...
> Our-C exited ...<<
```

// ==

當輸入是>>7

```
string str ;
char chArray[30] ;
cin >> chArray ;
str = "This is a fine day.\n" + chArray + "\n" ;
str = str + "Isn't it?\n" ;
cout << str ;
float hello ;
int x ;
hello=20*((x-hello)/34.5-17-(hello-x));
x=(x+8)/hello/(hello-8.0);
cout << "Value of 'hello' is now : " << hello << "\n" ;
Done();
```

<<

你的輸出應該是>>Our-C running ...
> Definition of str entered ...
> Definition of chArray entered ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Definition of hello entered ...
> Definition of x entered ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Our-C exited ...<<

// ==

// Prob. 5 of Proj. 2 ; Test case 2/3

當輸入是>>2

```
String str ;
string str;//comment should be skipped
char chArray[30] ;
cin >> chArray1 ;  // undeclared identifier
cin >> chArray ;
str = "This is a fine day.\n"
     + chArray
     + "\n"
     ;
str = str + "Isn't it?\n" ;
cout << str ;
str = "This is a fine day.\n"
     + "\n"
     + chArray1   // undeclared identifier within a statement
               // once an input error occurs, parsing
               // restarts from the next line
     ;     // a null statement is nevertheless "executable"
float hello ;
int x ;
hello = 20
       *          // comment should be skipped
       ( ( x - hello ) / 34.5
         - 17   // to the right is comment : + ( hello - x
         - ( hello - x )
       )
```

```
      ;
x=(x+8)/hello/(hello-8.0);
cout << "Value of 'hello' is now : "
    << hello
    << "\n"
    ;
hello = 20
      *
      ( ( x - hello135 ) / 34.5  // undeclared identifier
        - 17   // input restarts from this line
        - ( hello - x )
      )       // but then, this ')' is unexpected
      ;       // a null statement is fine though
Done();Done(how);Done(are);Done();
Done(you);
Done();
<<
```

你的輸出應該是>>Our-C running ...

```
> Line 1 : undefined identifier : 'String'
> Definition of str entered ...
> Definition of chArray entered ...
> Line 1 : undefined identifier : 'chArray1'
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Line 3 : undefined identifier : 'chArray1'
> Statement executed ...
> Definition of hello entered ...
> Definition of x entered ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Line 3 : undefined identifier : 'hello135'
> Line 3 : unexpected token : ')'
> Statement executed ...
> Our-C exited ...<<
```

```
// ============================================================
```

// Prob. 11 of Proj. 2 ; Test case 3/4

當輸入是>>3

```
int AddTwo( int x ) { return x + 2 ; } // comment
int AddFive( int x ) { int y ; y = AddTwo( x ) ; // comment
```

```
                    return y + 3 ; }          // comment
ListAllFunctions() ;
ListFunction( "AddFive" ) ;
int x ;
x = 100 ;
x = x + AddFive( x ) ;
if ( x > 200 )
  x = AddTwo( 300 ) ;
else
  x = x + AddFive( 200 ) + 5 ;
if ( AddTwo( x ) > 200 )
  x = 5 + AddThree( 300 ) ;
else
  x = x + AddFive( 200 ) + 5 ;
Done() ;
<<
```

你的輸出應該是>>Our-C running ...

```
> Definition of AddTwo() entered ...
> Definition of AddFive() entered ...
> AddFive()
AddTwo()
Statement executed ...
> int AddFive( int x ) {
  int y ;
  y = AddTwo( x ) ;
  return y + 3 ;
}
Statement executed ...
> Definition of x entered ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Line 2 : undefined identifier : 'AddThree'
> Line 1 : unexpected token : 'else'
> Statement executed ...
> Our-C exited ...<<
```

```
// ============================================================
```

```
// Prob. 13 of Proj. 2 ; Test case 3/4
```

當輸入是>>3

```
int test ;
char test ;  // re-define 'test'
```

```
int salary[30] ;

void InputSalary( int revenue[ 30 ] ) {
  int i ;
  i = 0 ;
  while ( i < 30 ) {
    cin >> revenue[ i ]
    i++ ;
    i=i+1;
  } // while ( i < 30 )
} // InputSalary()

void InputSalary( int revenue[ 30 ] ) {
  int i ;
  i = 0 ;
  while ( i < 30 ) {
    cin >> revenue[ i ] ;
    i++ ;
  } // while ( i < 30 )
} // InputSalary()

void Sort(int intArray[30]) {
  int i;
  i=0;
  while(i<29){
    int j;
    j=i;
    while(j<30){
      if(intArray[j]<intArray[i]){
      int temp;
      temp=intArray[i];
      intArray[i]=intArray[j];
      intArray[j] =temp;
      } // if intArray[ j ] < intArray[ i ]
      j++ ;
    } // while j < 30
    i++ ;
  } // while i < 29
} // Sort()

void InputSalary( int revenue[ 30 ] ) { // semantic error
  int i ;
  i = 0 ;
  while ( i < 30 ) {
    cout << revenue[ i ] ;
    i++ ;
  } // while ( i < 30 )
```

```
} // InputSalary()

void InputSalary( int revenue[ 30 ] ) {
  int i ;
  i = 0 ;
  while ( i < 30 ) {
    cin >> revenue[ i ] ;
    i++ ;
  } // while ( i < 30 )
} // InputSalary()

void OutputSalary( int revenue[ 30 ] ) {
  int i ;
  i = 0 ;
  while ( i < 30 ) {
    cout << revenue[ i ] ;
    i++ ;
  } // while ( i < 30 )
} // OutputSalary()

ListAllFunctions() ;

ListFunction( "Sort" ) ;

InputSalary( salary ) ;

Sort( salary ) ;

OutputSalary( salary ) ;

Done() ;
<<
```

你的輸出應該是>>Our-C running ...

```
> Definition of test entered ...
> New definition of test entered ...
> Definition of salary entered ...
> Line 7 : unexpected token : 'i'
> Line 1 : undefined identifier : 'i'
> Line 1 : unexpected token : '}'
> Line 1 : unexpected token : '}'
> Definition of InputSalary() entered ...
> Definition of Sort() entered ...
> New definition of InputSalary() entered ...
> New definition of InputSalary() entered ...
> Definition of OutputSalary() entered ...
> InputSalary()
```

```
OutputSalary()
Sort()
Statement executed ...
> void Sort( int intArray[ 30 ] ) {
  int i ;
  i = 0 ;
  while ( i < 29 ) {
    int j ;
    j = i ;
    while ( j < 30 ) {
      if ( intArray[ j ] < intArray[ i ] ) {
        int temp ;
        temp = intArray[ i ] ;
        intArray[ i ] = intArray[ j ] ;
        intArray[ j ] = temp ;
      }
      j++ ;
    }
    i++ ;
  }
}
Statement executed ...
> Statement executed ...
> Statement executed ...
> Statement executed ...
> Our-C exited ...<<
```

==========

Something to be aware of ...

Suppose the code user entered looks like this.

```
if ( ... )          // ... is a valid expression
  ...               // this ... is a valid statement
// ...              // this line is a comment
xxx                 // this line has an error
```

What is the line number of xxx?

Since the system (your program) has to see xxx before it can make the judgment of whether the IF-THEN-ELSE has ended or not, the comment line seems to belong to the IF-THEN-ELSE statement.

Anyway, this is a controversial situation (what is the line number of xxx?). Therefore, after detecting the existence of such code in the hidden test data, I have removed such cases so that the hidden test data no longer contain such code.