Practical -3 test cases

Test case 1:

File1.c:

```
int main() {
   int a = 5, 7H;
   // assign value
   char b = 'x';
   /* return
     value */
   return a + b;
}
```

Output:

```
PROBLEMS 4   TERMINAL   DEBUG CONSOLE   OUTPUT   PORTS   SEARCH TERMINAL OUTPUT                    Run: LexicalAnalyzer

PS C:\Users\YASH\Desktop\DLP cse313>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\YASH\AppData\Roaming\Code\User\workspaceStorage
a0774ad88195\redhat.java\jdt_ws\DLP cse313_7b443ede\bin' 'LexicalAnalyzer'
Lexical Analysis Output:
Keyword: int
Keyword: main
Punctuation: (
Punctuation: )
Punctuation: {
Keyword: int
Identifier: a
Operator: =
Constant: 5
Punctuation: ,
Constant: 7
Identifier: H
Punctuation: ;
Keyword: char
Identifier: b
Operator: =
Identifier: x
Punctuation: ;
Error: Unknown token '
Error: Unknown token '
Identifier: n
Error: Unclosed comment
Identifier: value
Operator: *
Operator: /
Keyword: return
Identifier: a
Operator: +
Identifier: b
Punctuation: ;
Punctuation: }

Symbol Table:
a : Identifier
b : Identifier
H : Identifier
x : Identifier
value : Identifier
n : Identifier
PS C:\Users\YASH\Desktop\DLP cse313>
```

Test case -2

File2.c:

```
/* salary calculation*/
void main( ) {
   long int bs , da , hra , gs;
//take basic salary as input
scanf("%ld",&bs);
//calculate allowances
da=bs*.40;
hra=bs*.20;
gs=bs+da+hra;
// display salary slip
printf("\n\nbs : %ld",bs);
printf("\nda : %ld",da);
printf("\nhra : %ld",hra);
printf("\ngs : %ld",gs);
}
```

Output:

```
PS C:\Users\YASH\Desktop\DLP cse313>  c:; cd 'c:\Users\YASH\Desktop\DLP cse313'; & 'C:\Program Files\Java\jdk-21\bin\jav
\AppData\Roaming\Code\User\workspaceStorage\cd61bebce88762841ddba0774ad88195\redhat.java\jdt_ws\DLP cse313_7b443ede\bin'
Lexical Analysis Output:
Keyword: void
Keyword: main
Punctuation: (
Punctuation: )
Punctuation: {
Identifier: long
Keyword: int
Identifier: bs
Punctuation: ,
Identifier: da
Punctuation: ,
Identifier: hra
Punctuation: ,
Identifier: gs
Punctuation: ;
Keyword: scanf
Punctuation: (
String: "%ld"
Punctuation: ,
Operator: &
Identifier: bs
Punctuation: )
Punctuation: ;
Identifier: da
Operator: =
Identifier: bs
Operator: *
Constant: 40
Punctuation: ;
Error: Unknown token .
Identifier: hra
Operator: =
Identifier: bs
Operator: *
Constant: 20
Punctuation: ;
Error: Unknown token .
Identifier: gs
```

```
Error: Unknown token .
Identifier: gs
Operator: =
Identifier: bs
Operator: +
Identifier: da
Operator: +
Identifier: hra
Punctuation: ;
Keyword: printf
Punctuation: (
String: "\n\nbs : %ld"
Punctuation: ,
Identifier: bs
Punctuation: )
Punctuation: ;
Keyword: printf
Punctuation: (
String: "\nda : %ld"
Punctuation: ,
Identifier: da
Punctuation: )
Punctuation: ;
Keyword: printf
Punctuation: (
String: "\nhra : %ld"
Punctuation: ,
Identifier: hra
Punctuation: )
Punctuation: ;
Keyword: printf
Punctuation: (
String: "\ngs : %ld"
Punctuation: ,
Identifier: gs
Punctuation: )
Punctuation: ;
Punctuation: }

Symbol Table:
bs : Identifier
hra : Identifier
gs : Identifier
da : Identifier
long : Identifier
PS C:\Users\YASH\Desktop\DLP cse313>
```

Test case-3

File3.c

```
//function prototype
void add ( int , int );
void main( )
{
int a , b;
a = 10;
b = 20;
```

```
// function call
add ( a , b );
}
void add ( int x , int y )
{
return x + y;
}
```

Output:

```
PS C:\Users\YASH\Desktop\DLP cse313>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetails
88195\redhat.java\jdt_ws\DLP cse313_7b443ede\bin' 'LexicalAnalyzer'
Lexical Analysis Output:
Keyword: void
Identifier: add
Punctuation: (
Keyword: int
Punctuation: ,
Keyword: int
Punctuation: )
Punctuation: ;
Keyword: void
Keyword: main
Punctuation: (
Punctuation: )
Punctuation: {
Keyword: int
Identifier: a
Punctuation: ,
Identifier: b
Punctuation: ;
Identifier: a
Operator: =
Constant: 10
Punctuation: ;
Identifier: b
Operator: =
Constant: 20
Punctuation: ;
Identifier: add
Punctuation: (
Identifier: a
Punctuation: ,
Identifier: b
Punctuation: )
Punctuation: ;
Punctuation: }
Keyword: void
Identifier: add
Punctuation: (
Keyword: int
Identifier: x
Punctuation: ,
Keyword: int
Identifier: y
Punctuation: )
Punctuation: {
Keyword: return
Identifier: x
Operator: +
```

```
Identifier: y
Punctuation: )
Punctuation: {
Keyword: return
Identifier: x
Operator: +
Identifier: y
Punctuation: ;
Punctuation: }

Symbol Table:
add : Identifier
a : Identifier
b : Identifier
x : Identifier
y : Identifier
PS C:\Users\YASH\Desktop\DLP cse313>
```