

Assignment 8: C String Literals and Comments

Complete assignment due: Thurs., October 23rd, 9:00 p.m.

String Literals: (40 points)

Write a program named `literal.c`. The program will read from standard input and write to standard output. The input will be a C program file. The output will be the string literals that are in the C file.

When your program finds the start of a C string literal, print the entire string literal, including the quotation marks. Start each string literal on a new line, with the leading quotation mark as the first character on the line.

After the string literals have been printed, optionally print a count of the number of literals found, and a count of the number of lines that were in the C file.

Your program will take two optional command-line arguments:

- `-s` Print the number of string literals found in the input
- `-l` Print the number of lines found in the input

Some examples:

```
wolf-> cat test1.c
#include <stdio.h>
#include <stdlib.h>

int
main( int argc, char *argv[] )
{
    // one-line comment
    char *s = "initialization";

    /* traditional comment */

    printf("Hello world %s\n", s);

    /* a multi-line
     * "comment going"
     * on here
     */

    printf("one literal here %s\n", "another one here");

    /* two comments */ int a = 7; /* on the same line */
    printf("a = %d\n", a); // printed on this line

    return 0;
} /* main */
lectura-> ./literal < test1.c
```

```

"initialization"
"Hello world %s\n"
"comment going"
"one literal here %s\n"
"another one here"
"a = %d\n"
lectura-> ./literal -s < test1.c
"initialization"
"Hello world %s\n"
"comment going"
"one literal here %s\n"
"another one here"
"a = %d\n"
Count of string literals: 6
lectura-> ./literal -l < test1.c
"initialization"
"Hello world %s\n"
"comment going"
"one literal here %s\n"
"another one here"
"a = %d\n"
Count of lines: 25
lectura-> ./literal -l -s < test1.c
"initialization"
"Hello world %s\n"
"comment going"
"one literal here %s\n"
"another one here"
"a = %d\n"
Count of string literals: 6
Count of lines: 25
lectura-> ./literal -x < test1.c
./literal: -x: unknown option
Usage: ./literal [-s] [-l]

```

Errors:

Your program should detect the following errors:

- Invalid command-line arguments: anything other than **-s** or **-l**.

Turnin:

Use the **turnin** program to turn in all the files for your program. The command is:

```
turnin 352assign8 literal.c
```

Comments: (60 points)

Write a C program named **comment.c**. The program will read from standard input and write to standard output. The input will be a C program file. The output will be all of the one-line comments found in the C file, followed by all the multi-line comments found in the C file.

For `//`-style comments: When your program finds the start of a one-line comment, print the entire comment, including the two slash marks. Start each comment on a new line, with the leading slash as the first character on the line.

For `/*`-style comments: Print the comment starting on a new line. The first character on the line will be the leading slash from the comment. The comment will then follow, possibly using multiple lines. Thus, when you find a newline character in the comment, simply output the newline character.

Note that you will only be able to read the file once, since we are using standard input. You will need to find both types of comments simultaneously. You will be able to print the `//`-style comments as you find them. However, the `/*`-style comments will have to be saved so they can be printed later. Since we have not covered dynamically-allocated memory yet, we guarantee that the input file will not contain more than 100,000 characters that are part of `/*` style comments.

After the string literals have been printed, optionally print a count of the number of literals found, and a count of the number of lines that were in the C file. This will give a rough indication of how much of the C file is composed of string literals.

Your program will take two optional command-line arguments:

- `-1` (the number one) Print the number of `//`-style comments found in the input
- `-c` Print the number of `/*`-style comments found in the input
- `-l` (the letter “l”) Print the number of lines found in the input

Some examples: These examples use the same `test1.c` file shown in the examples for `literal.c`.

```
lectura-> ./comment < test1.c
// one-line comment
// printed on this line
/* traditional comment */
/* a multi-line
 * "comment going"
 * on here
 */
/* two comments */
/* on the same line */
/* main */
lectura-> ./comment -1 < test1.c
// one-line comment
// printed on this line
/* traditional comment */
/* a multi-line
 * "comment going"
 * on here
 */
/* two comments */
/* on the same line */
```

```

/* main */
Count of //-style comments: 2
lectura-> ./comment -1 -c < test1.c
// one-line comment
// printed on this line
/* traditional comment */
/* a multi-line
   * "comment going"
   * on here
   */
/* two comments */
/* on the same line */
/* main */
Count of //-style comments: 2
Count of /*-style comments: 5
lectura-> ./comment -c -1 -1 < test1.c
// one-line comment
// printed on this line
/* traditional comment */
/* a multi-line
   * "comment going"
   * on here
   */
/* two comments */
/* on the same line */
/* main */
Count of //-style comments: 2
Count of /*-style comments: 6
Count of lines: 25
lectura-> ./comment -1 -c -z
./comment: -z: unknown option
Usage: ./comment [-1] [-c] [-1]
lectura->

```

Errors:

Your program should detect the following errors:

- Invalid command-line arguments: anything other than **-1**, **-c**, or **-1**.

Turnin:

Use the **turnin** program to turn in all the files for your program. The command is:

```
turnin 352assign8 comment.c
```

Final Summary:

There are two files that you turnin for full credit on this assignment:

```
literal.c
comment.c
```