Project 1 (5%) - Due Aug 16th

Overview of Project 1

In this project, you develop a C99 C program named `draw1a` (that's the number one in 1a, not the letter l). It will read an input file, recognize the type of lines in the input, and count how often each type occurs. After reading all input, it will write some statistics to standard output (`stdout`). In Linux, output sent to `stdout` shows up on your monitor by default (but it can be easily be redirected e.g. to a file).

Input/Output and Behavior Specification

Your C99 C program, called `draw1a`, also referred to as *your program*, takes one command-line argument that specifies the filename of the input file. An example of how your program is invoked from the command-line is:

```
% ./draw1a ./inputfile.1
```

When `draw1a` starts up, it prints a two-line **start-up message** to `stdout`. It consists of

1. a message stating how and when your program was started, and
2. its command-line argument

It is printed in the following format:

```
./draw1a started on Wed Jul 2 21:08:40 MST 2010

Input file: ./inputfile.1
```

See the Hints section for some help doing this.

inputfile.1 is an example of a valid input file. All input files are plain text files written in a "little language" for graphics.

Each line starts with one of the following 8 keywords: `Image, lineTo, End, print, draw, translate, child, #` (for comments).

Each line is terminated by a newline character `'\n'`.

`draw1a` should read the input one line at a time, identify the keyword at the start of the line, and count how often each keyword occurs in the input file. As output, `draw1a` prints statistics to stdout, listing how often each type of line was encountered in the input. For the sample inputfile.1 (see the sample input section below), the complete output should be (your printed date will differ):

```
./draw1a started on Wed Jul 2 21:08:40 MST 2010

Input file: ./inputfile.1

2 Image definition(s)

6 lineTo command(s) within Figures

2 End command(s)

2 print command(s)

4 draw command(s)

1 translate command(s)

3 child command(s)

6 comment(s)
```

Use the same order of keywords and the same text in this example. All your output should be sent to `stdout`. Print 0 if no lines of a certain type were found, e.g.

```
0 comment(s)
```

The statistics that you output to `stdout` have to match our expected output when compared by the Linux command `diff -b` (see Resources section). Your program will be tested on many different input files. For this project, it is enough to recognize the keyword in each line.

Tokens

Each input line can be thought of as built of tokens. Each token is a sequence of characters. Tokens are separated from each other by one or more whitespace characters. Tokens do not contain any whitespace within the token. For example, in the line

```
Image A 100 100 =
```

There are five tokens,

```
1. Image
```

```
2. A
```

```
3. 100
```

```
4. 100
```

```
5. =
```

For this project, only one token per line, the single keyword at the beginning of each line, is relevant. The `scanf` family of functions (e.g. `fscanf`, `sscanf`) with format `%s` can be used to parse tokens as text strings in C.

Sample Input

An example of a valid input file `inputfile.1` is:

```
# Comments look like this
Image A 100 100 =
# Image A (2 diagonal line segments)
lineTo 100 100
lineTo -100 100
End Image A
# ############################
Image B 10 10 =
# Image B (a square)
lineTo 390 0
lineTo 0 390
lineTo -390 0
lineTo 0 -390
End Image B
# ############################
print A
print B
draw A
draw B
translate A 20 30
draw A
draw B
child pause 2
child clearScreen
# child end below is last line in input file
child end
```

Resources

Study the following resources as needed:

- standard C library functions for files, strings
- standard Unix/Linux functions popen, pclose

Guarantees about Test Input

For the project 1, your code will only be tested on input which is valid according to the specifications. Furthermore, it will obey the following restrictions:

- The input file's name will be no more than 128 characters.
- Input lines will be no more than 256 characters (including the newline).
- Each line starts with one of the 8 keywords. There will be no leading whitespace on a line.

Hints

- To print the start-up message, use the following hints:
    - Use `fflush` right before you call `system` (see immediately below) to ensure the order of your output is correct. Use it to flush all open output streams. See its `man` page for more information.
    - Use the line
    - `system("date");`
    - to call the Linux `date` command, which prints out the current date and time. See the `system` man page for the header file to include.