CS2123 Program #1 Infix to Postfix (30 points)

Convert expressions that are in infix notation to postfix notation. The expressions might contain parentheses and these operators: =, ONLY, NEVER, AND, OR. In program #2, you will evaluate that postfix expression.
Examples:

Infix	Postfix
RECIT = N	RECIT N =
RECIT = Y	RECIT Y =
PROF = CLARK	PROF CLARK =
PROF NEVER CLARK	PROF CLARK NEVER
PROF ONLY CLARK	PROF CLARK ONLY
PROF NEVER CLARK AND DEPT = CS	PROF CLARK NEVER DEPT CS = AND
RECIT = Y AND (PROF = CLARK OR PROF = GIBSON)	RECIT Y = PROF CLARK = PROF GIBSON = OR AND

Precedence rules

- Evaluate contents in () before others
- =, NEVER, ONLY - high precedence
- AND, OR - lower precedence

Driver Program and your function in separate files

For this program, I provided a driver program which will invoke your function to convert from infix to postfix. **Your code must be placed in a separate file**.

Why did I provide a driver program?

- Reduces the amount of code you have to write. This allows you to focus on the most important points.
- Teaches you what it is like to use code written by someone else.
- Shows you the value of programming standards.
- Provides a design which makes it easier for you to do this program. It is very helpful to have **addPostfixOut**, **categorize**, and **getToken**.

Why must you put your code in a separate file?

- In the *real* world, many programmers work on the same program. It is significantly easier to manage code when it is separated into many files.

Some Files for your use

cs2123p1.h - include file for Program #1. Please review this.

cs2123p1Driver.c - driver program that I provided. It has several useful routines to help reduce your effort. Please review this code.

p1Input.txt - input file containing infix expressions, one per text line.

p1Extra.txt - input expressions, one per text line. This is used for extra credit. Note that it also contains the expressions in p1Input.txt.

Makefile.txt - rename to simply Makefile, this will help you compile and link your code.

Note: some Internet browsers won't allow .c and .h files to be downloaded. To avoid that problem, I changed the file names to cs2123p1_h.txt and cs2123p1Driver_c.txt. Please rename them.

Your code

- Create your code in **p1*abc123*.c** (replace *abc123* with your abc123 ID). It must not contain the code from the driver!! Based on what the driver calls, you need to create (at least) this function:

```
int convertToPostfix(char *pszInfix, Out out)
        It returns 0 if it converted successfully.  Otherwise, it returns a value which
        indicates an error in the infix expression (e.g., missing left paren, missing
        right paren).
        It populates the out array using the addPostfixOut function (provided in the
        driver).
```

  For **modularity**, you will need to divide `convertToPostfix` into multiple functions.
- To compile the driver, your code, and create an executable, use the **make** utility (see below).

Error Handling

- Your code must handle errors like the following:
  - Missing "("
  - Missing ")"
  - Additional errors are handled for extra credit.
- When an error is encountered, your code for convertToPostfix should return a non-zero value to the driver. Your program must not terminate.

Requirements

1. Your code must be written according to my programming standards.
2. You should not have to modify cs2123p1Driver.c. If you want to modify it, please see me before you do that.
3. See below for what to turn in via BlackBoard.
4. The output should show a warning for each poorly formed expression, but do not terminate the program (simply skip to the next expression).
5. Make certain you free up allocated memory (e.g., stack).
6. **Modularity matters**. You probably need to code more than just the convertToPostfix function.

Hint

The functions **getToken** and **categorize** are provided in the cs2123p1Driver.c. These can greatly simplify your code.

Meaning of the = operator

- The "=" operator is interpreted as "is at least". "PROF = CLARK" means the PROF attribute type must have a value of at least CLARK. It may have other values in the data for a particular course.
- This meaning of "=" is much more important for program #2.

Extra Credit (10 pts + 100 / N)

- Extra credit is all or nothing. Your program must meet ALL requirements.
- You will **not receive extra credit** if your program is turned in **late**.
- N = the number of people who get it completely correct. For example, if only 4 people get the extra credit completely correct, they each receive an extra 35 points (10 + 100 / 4).
- The following errors must be detected:
  - Missing "(" -- must also be handled for normal credit
  - Missing ")" -- must also be handled for normal credit
  - Missing operator
  - Missing operand
- You must also turn in your output for the extra credit input file.
- If your program does not follow my programming standards, it will not receive extra credit.

Compiling Using the make Utility

Before using the make utility, you must:

- Download the Makefile.txt file and rename it simply **Makefile** (i.e., remove the .txt suffix)
- Edit Makefile to change p1abc123.o with your abc123 id. Note that you are changing a **.o** file reference. The make utility will automatically reference your .c file if you properly name the p1abc123.o file:

```
# Define the machine object files for your program
OBJECTS = p1abc123.o cs2123p1Driver.o
# Define your include file
INCLUDES = cs2123p1.h

# make for the executable
p1: ${OBJECTS}
```

```
	gcc -g -o p1 ${OBJECTS}

# Simple suffix rules for the .o
%.o: %.c ${INCLUDES}
	gcc -g -c $<

# Clean the .o files
clean:
	rm -f ${OBJECTS}
```

Based on the rules in the Makefile, when you tell make to make your executable, it will **automatically compile** your .c file and the driver .c file. (For more information about the **make** utility, click here.)

```
make p1
```

Executing the p1 executable:

```
./p1 < p1Input.txt
```

Turn in LastNameFirstName**.zip** containing

- Your include file (if it changed)
- Your p1*abc123*.c file.
- Your Makefile (since you changed it for your p1*abc123*.c). The TA/grader will use your Makefile to make the code.
- Your output based on the data provided. If you did the extra credit, turn in the output for just the extra credit
- Also, provide a note to the TA on whether you did the extra credit.