PROBLEM DEFINITION

Write a program in C/C++ that translates a pattern specification provided as an command line argument into a regular expression, processes lines of input text received from stdin using that regular expression to qualify matches, and finally writes each matching input line to stdout. Each line of input is terminated by the newline character '\n', and the program should terminate when it receives EOF.

The program should be executable as follows:

$ cat input.txt | program "is this message %{0} ballpark %{1S3}" > output.txt

The program should be structured to parse the pattern specification and generated a regular expression which expresses the pattern. The program should then use the libpcre library (www.pcre.org) available on linux and Windows systems, to execute matches to qualify each line of text received on stdin.

PATTERN SPECIFICATION

A pattern is a text string, delimited with *token capture sequences* which identify the variable text extracted from the message. A *token capture sequence* is represented as a percent sign '%' character followed by a '{' character, non-negative integer, an optional token capture modifier, and finally a '}' character. The non-negative integer denotes the index into the token list associate with the rule to which the pattern belongs. A simple *token capture sequence* would be written as "%{0}" and "%{25}", and will capture any amount of text which occurs between the adjacent text literals. For example of a pattern with only simple token escape sequences follows:

"foo %{0} is a %{1}"

which would match the following text strings:

"foo blah is a bar"
"foo blah is a very big boat"

but would not match the following text strings:

"foo blah is bar"
"foo blah"
"foo blah is"

A *token capture modifier* specifies special handling of the token capture in order to differentiate otherwise ambiguous patterns. There are two types of token capture modifiers: space limitation (S#) and greedy (G). A space limitation modifier, specifies a precise number of whitespace characters which must be appear between text literals in order to match the associated *token capture sequence*. For example, the *token capture*

sequence %{1S2} specifies token one (1), with a space limitation modifier of exactly two (2) spaces. For example, the pattern:

"foo %{0} is a %{1S0}"

which would match the following text string:

"foo blah is a bar"

but would not match the following text strings:

"foo blah is a very big boat"
"foo blah is bar"
"foo blah"
"foo blah is"

The whitespace modifier is intended to be used to limit possibly ambiguous matches, as in the example above, or in cases where two *token capture sequences* occur adjacent within a pattern without intervening literal text. For example, the pattern:

"the %{0S1} %{1} ran away"

would match the text string

"the big brown fox ran away"

which would capture "big brown" for token %{0S1}, and "fox" for token %{1}.

A greedy *token capture modifier* specifies special handling of the token capture in order to differentiate patterns which are ambiguous due to repetitions of token value text that also occurs in the literal text of the pattern. The greedy capture modifier captures as mush as possible text between preceding and following string literals. For example, the pattern:

"bar %{0G} foo %{1}"

would match the text string:

"bar foo bar foo bar foo bar foo"

and capture "foo bar foo bar" for token specifier %{0G} and "bar foo" for token specifier %{1}.

PROGRAM EVALUATION

The program should be submitted in source form, and should use what you consider good coding standards. Comments should be used within the code to explain interesting or tricky segments that would help someone else who needs to maintain the code. Clarity and correctness are of paramount importance, with efficiency following those.