Project 1

Overview

You may work with a partner for this project. Each of you will submit the following:

- Completed rgrep.c
- Any test files you used to check if your solution is correct
- Name of your partner in the submission box (if any)

Background

`grep` is a UNIX utility that is used to parse or search through strings for a particular pattern. The strings can be either put in through the console or in text files. It is a very convenient way to look for basic patterns or ones with wildcards. It is fairly complicated to support the full character set that `grep` is capable of. So in this project, you will only implement a restricted grep, `rgrep`.

Examples using the files in your project bundle are as follows. First you can see the contents of the basic test file `short.txt`.

```
$ cat short.txt
aa
aah
aahed
aahing
aahs
aardwolf
aardwolves
aas
aasvogel
aasvogels
abaci
aback
abacus
abacuses
zyme
zymogens
zymologies
zymurgies
zymurgy
zzaabb
sF1xxO?
t.bO?T1
a2\W4pH
DTJg2gQ
qkp9H9M
TMLBIPV
Ih?Lutl
bB0hy1w
jYed9FK
qQqMfDl
.?as..\?
```

Simple call with basic pattern aa results as follows:

```
$ ./rgrep 'aa' < short.txt
aa
aah
aahed
aahing
aahs
aardwolf
aardwolves
aas
aasvogel
aasvogels
zzaabb
```

Notice that the last line has a substring containing “aa” whereas the beginning examples all starts with “aa”. So be sure to detect for substrings. Next example looks for presence of ‘?’ as a character but not a wildcard.

```
$ ./rgrep '\?' < short.txt
sF1xxO?
t.bO?T1
Ih?Lutl
.?as..\?
```

Find all the strings with g in them:

```
$ grep 'g' short.txt
aahing
aasvogel
aasvogels
zymogens
zymologies
zymurgies
zymurgy
DTJg2gQ
```

Find strings that are at least 7 characters long:

```
grep '.......' short.txt
aardwolf
aardwolves
aasvogel
aasvogels
abacuses
zymogens
zymologies
zymurgies
zymurgy
sF1xxO?
t.bO?T1
a2\W4pH
DTJg2gQ
qkp9H9M
TMLBIPV
Ih?Lutl
bB0hy1w
```

```
jYed9FK
qQqMfDl
.?as..\?
```

Complete list of the wildcard characters to be implemented are here

. (period)	Matches any character.
+ (plus sign)	The preceding character will appear one or more times.
? (question mark)	The preceding character may or may not appear in the line.
\ (backslash)	"Escapes" the following character, nullifying any special meaning it has

Additionally, here are examples and how to use the wildcard characters

a+	Matches a, aa, aaaaa or really any number of a's more than one
.+	Matches any non-empty String
\\+	Matches a string of \'s
a?b+	Matches ab, b, abbb or any amount of b following optional a.
\?	A question mark must appear in the line
they?re	Matches a line that contains either the string "theyre" or the string "there"
h.d..?n	Matches lines that contain substrings like "hidden", "hidin", "hbdwen", "hadbn", etc.
cu\.?t	Matches lines that either contain the substring "cut" or "cu.t"

Only the legal strings and patterns will be checked. You are expected to create more testing patterns on your own and test your implementation. Skeleton code is given to you and you should not modify it.

Getting started

Download the zip file that contains all the files. To compile, simply typing make will build everything you need:

```
make
```

Once compiled, the executable can be invoked as follows:

```
./rgrep pattern
```

your implementation should go inside the function `rgrep_matches()`, which returns true if and only if the string matches the pattern. You can add or use any helper functions or data structures as needed to solve this problem. The skeleton will automatically read each line of code from the input and runs it against the pattern. You job is to just identify whether each line matches.

Testing

For sanity check, Makefile has a bunch of basic tests it runs to see if you implemented the basic or wildcard characters correctly. The list is purposely not complete and definitely not covering the corner cases. The expectation for your testing strategy is to uncover those harder patterns and implement accordingly.

```
$ make check
```

When running the sanity checks, make sure to turn off **all prints** you may have used for debugging. It is checking your output compared with the expected output. So if you have any prints at all then it will change the output. Suggestion is to use a debugging flag to turn on/off prints while you are working through out. Obviously when you submit your final code, you should turn off debugging.

Submission

- Completed rgrep.c
- All the additional test files