Match Assignment for u0682219

The goal of this assignment is to give you some practice with C on a problem that is algorithmically straightforward but difficult to get right on a first try. With any luck, you'll have a bug along the way, and then you get to practice using GDB.

Your task is to implement a program that takes command-line arguments and checks whether each argument matches a particular pattern. The pattern to check is determined by a command-line flag: `-a`, `-b`, or `-c`, where the default mode is `-a`. The patterns for each flag are described below.

By default, your program should print "yes" (always lowercase, no extra space) for each matching argument and "no" (always lowercase, no extra space) for each non-matching argument, each on a separate line. If the `-t` flag is provided, then your program should instead print nothing for non-matching argument and it should print a converted value for each matching argument. Each pattern mode has an associated conversion, as described below.

At most one of `-a`, `-b`, or `-c` will be provided. A `-t` flag can be provided before or after any of `-a`, `-b`, and `-c`. All flags (i.e,. up to two of them) precede other command-line arguments. A non-flag argument never starts with `-`. All arguments are in ASCII.

See the end of this page for example command-line uses of your program. *Neither the examples there nor the examples for indvidual patterns are meant to be exhaustive.*

Constraints: Your implementation must be in ANSI C, so that it compiles without errors or extra flags on the CADE `lab1-`*n*`.eng.utah.edu` machines using the default `gcc`. Your implementation must not depend on any libraries other than the standard C library; please note that while `"regex.h"` and its associated functions are POSIX-standard, they are not part of ANSI C and therefore not allowed in this assignment. Also, yor implementation must not use multiplication, division, or modulo operators. (Bitwise operations offer alternatives to those operations.) You should hand in one `matchlab.c` file so that `gcc -o matchlab matchlab.c` on a CADE lab machine builds your program as `matchlab`; your program should work the same with and without debugging and optimization options such as `-g` or `-O2`. Hand in via Canvas.

Important! Make sure the uID listed at the top of the page is correct, because this assignment (and its correct answer) is uID-specific.

Where the descriptions below say "even position" or "odd position", counting is from 0, as in C. So, a first character is in an even position.

`-a` mode

Match a sequence of the following, with nothing else before, after, or in between:

- between 4 and 5 repetitions (inclusive) of the letter "g";
- exactly one "=";
- any odd number of repetitions of the letter "z";
- exactly one "_"; and
- between 1 and 3 (inclusive) decimal digits.

For matches, perform the following conversion:

- reverse all of the characters.

Example arguments that match, followed by their conversions:

- `gggg=z_185`
  → `581_z=gggg`
- `gggg=z_552`
  → `255_z=gggg`
- `gggg=z_838`
  → `838_z=gggg`
- `gggg=zzzzz_533`
  → `335_zzzzz=gggg`
- `gggg=zzzzzzzzz_969`
  → `969_zzzzzzzzz=gggg`
- `gggg=zzzzzzzzz_649`
  → `946_zzzzzzzzz=gggg`
- `gggg=zzz_425`
  → `524_zzz=gggg`
- `gggg=zzzzzzz_621`
  → `126_zzzzzzz=gggg`
- `gggg=z_26`
  → `62_z=gggg`
- `gggg=zzz_919`
  → `919_zzz=gggg`

Example arguments that do not match:

- `.gggg=z_185`
- `gggg=z_185.`
- `gggg:zzzzzzzzz_860`
- `gggg_zzz_756`
- `gggg=zzzzzzzzz:805`
- `gggg=wwwwwwww:532`
- `jjj:zzzzz_887`
- `gggg,zzzzzzzzz_665`
- `gggg=ssss_286`
- `gggg_z,211`
- `jjj=zzzzz:896`
- `gggg=yyy_52`
- `gggg_492=z`
- `_gggg=483zzzzzzzzz`

`-b` mode

Match a sequence of the following, with nothing else before, after, or in between:

- any even number (including zero) repetitions of the letter “m”;
- exactly one “:”;
- an odd number of uppercase letters — call this sequence *X*;
- 1 or more repetitions of the letter “r”;
- exactly one “:”;
- the same characters as the even-positioned characters in *X*; and
- between 1 and 3 (inclusive) decimal digits.

For matches, perform the following conversion:

- add one “H” after each “E”.

Example arguments that match, followed by their conversions:

- `mmmm:Crrrrrrrr:C232`
  → `mmmm:Crrrrrrrr:C232`
- `mmmmmm:HEGrrr:HG538`
  → `mmmmmm:HEHGrrr:HG538`
- `:BCCrr:BC173`
  → `:BCCrr:BC173`
- `mmmmmmmm:BDFrrrrrrr:BF252`
  → `mmmmmmmm:BDFrrrrrrr:BF252`
- `mm:BBArrrrr:BA649`
  → `mm:BBArrrrr:BA649`
- `mmmm:DBBDBrrrr:DBB703`
  → `mmmm:DBBDBrrrr:DBB703`
- `mmmm:EBBrrrrrrrrr:EB965`
  → `mmmm:EHBBrrrrrrrrr:EHB965`
- `mmmm:CDGrrrrr:CG503`
  → `mmmm:CDGrrrrr:CG503`
- `mmmmmmmm:Brrrr:B997`
  → `mmmmmmmm:Brrrr:B997`
- `mmmmmmmm:Errr:E375`
  → `mmmmmmmm:EHrrr:EH375`

Example arguments that do not match:

- `.mmmm:Crrrrrrrr:C232`
- `mmmm:Crrrrrrrr:C232.`
- `llllllll:Crrr,80`
- `mm,Errr=E510`
- `mmmm:BDGADrrrr,DA947`
- `mmmmmmmm:EFDxxxxx:ED274`
- `mm:AFDq:AD119`
- `,Czz,C725`
- `mmmmmm_EGDEFrrrr:EDF754`
- `mm=Grrr:G722`
- `mm:Arrr=A503`
- `mm:Dqqq:D601`
- `:rrrrr:333DDEmmmmmmDE`
- `mmmmmmrrrrrrr446::GG`

`-c` mode

Match a sequence of the following, with nothing else before, after, or in between:

- 4 or more repetitions of the letter “i”;
- exactly one “,”;
- an odd number of uppercase letters — call this sequence *X*;
- any odd number of repetitions of the letter “s”;
- exactly one “_”;
- between 1 and 3 (inclusive) decimal digits; and
- the same characters as *X* repeated 3 times.

For matches, perform the following conversion:

- remove every “F” and “C”.

Example arguments that match, followed by their conversions:

- `iiiiiiiii,Fsss_791FFF`
  → `iiiiiiiii,sss_791`
- `iiiiiiii,Bsss_148BBB`
  → `iiiiiiii,Bsss_148BBB`
- `iiiiiiii,AEEsss_781AEEAEEAEE`
  → `iiiiiiii,AEEsss_781AEEAEEAEE`
- `iiiiii,GDEEGsss_854GDEEGGDEEGGDEEG`
  → `iiiiii,GDEEGsss_854GDEEGGDEEGGDEEG`
- `iiiiii,FGGFCsss_391FGGFCFGGFCFGGFC`
  → `iiiiii,GGsss_391GGGGGG`
- `iiiiiiii,Bsssssss_219BBB`
  → `iiiiiiii,Bsssssss_219BBB`
- `iiiiiiii,HBFsssssss_857HBFHBFHBF`
  → `iiiiiiii,HBsssssss_857HBHBHB`
- `iiiiiiiii,FGCCCsssssss_613FGCCCFGCCCFGCCC`
  → `iiiiiiiii,Gsssssss_613GGG`
- `iiiiii,Hssss_6HHH`
  → `iiiiii,Hssss_6HHH`
- `iiiiii,CFGssss_302CFGCFGCFG`
  → `iiiiii,Gssss_302GGG`

Example arguments that do not match:

- `.iiiiiiiii,Fsss_791FFF`
- `iiiiiiiii,Fsss_791FFF.`
- `iiiiii,EAGsss_572GAE`
- `jjjjjjj,GHHssss_698GHHGHHGHH`
- `iiiiiiiii,Dsss=436DDD`
- `hhh,Gsssssss_552GGG`
- `iiiiii,Cyyyyyyy_309CCC`
- `iiiiiii,FBGGBssss_119FGB`
- `cccc:Gnnnnnnnn_419GGG`
- `d,HDFDHvvvvvvv_893HDFDHHDFDHHDFDH`
- `cc,BEDABuu_513BEDABBEDABBEDAB`
- `iiiii:GEGssss_334GEGGEGGEG`
- `iiiiiiiiiCHH,_sssssssCHHCHHCHH433`
- `318sssHHDiiiiiiHHDHHDHHD,_`

Examples

Assuming that your program is created as `./matchlab`, here are some example command-line uses (where “$” is a command-line prompt). We expect your program to match this output exactly.

```
$ ./matchlab gggg=zzzzzzz_89
yes
$ ./matchlab gggg=yyy:733
no
$ ./matchlab gggg=zzzzzzz_89 gggg=yyy:733
yes
```

```
no
$ ./matchlab -t gggg=zzzzzzz_89
98_zzzzzzz=gggg
$ ./matchlab -t gggg=yyy:733 gggg=zzzzzzz_89
98_zzzzzzz=gggg
$ ./matchlab -t gggg=zzzzzzz_89 gggg=zzzzzzz_203
98_zzzzzzz=gggg
302_zzzzzzz=gggg
$ ./matchlab -t -b mmmmmmmm:ABDDGrrrrrrrrr:ADG696
mmmmmmmm:ABDDGrrrrrrrrr:ADG696
$ ./matchlab -c -t iiiiiii,Dsss_922DDD
iiiiiii,Dsss_922DDD
$ ./matchlab -c iiiiiii,Dsss_922DDD
yes
```