

Forum:[Hack Forums](#)

Thread: [Challenge: Strings](#)

WOLGA - January 21, 2016, 5:29 pm

Challenge by Qwertyuioppasdfghjklzxc:

Given two alphanumeric strings, check to see if one can be rearranged case-insensitively to form the other. Spaces and quotes should not count for checking.

Example:

```
Code $ check.bat "abcdefg" "gfedcba"
yes
```

```
$ check.bat "abc123" "312def"
no
```

```
$ check.bat "blah blah" "blahblah"
yes
```

```
$ check.bat "123123" "123"
no
```

If you need clarification please ask.

GL1TCH3D - January 21, 2016, 7:00 pm

I'm assuming that it's case insensitive?

jahwi - January 21, 2016, 7:01 pm

this one I'll enjoy doing...Will do this soon.

WOLGA - January 21, 2016, 7:12 pm

(Jan 21, 2016 08:00 PM)GL1TCH3D Wrote: [\[CITE\]](#) I'm assuming that it's case insensitive? Yes, that is correct, and thank you for asking. I have updated the OP.

GL1TCH3D - January 21, 2016, 8:11 pm

<http://pastebin.com/ywSzC6Y5>

Should work

White Light - January 22, 2016, 12:41 am

Rearrange as in no duplication of any characters, correct?

```
If that's the case... \(Click to View\)Code @echo off
setlocal enabledelayedexpansion
set s1=%~1
set s2=%~2
```

```
set #=%s1%
set l=0
```

```

for %%A in (64 32 16 8 4 2 1) do if not "!#:~%%A!"==" set #=!#:~%%A!&set /a l+=%%A
if not "!s2:~%l%!"=="!s2:~%l%,1!" goto :no
for /l %%A in (0,1,%l%) do (
    for %%a in ("!s1:~%%A,1!") do if "!s2:%%~a!="=="!s2!" goto :no
    for %%a in ("!s2:~%%A,1!") do if "!s1:%%~a!="=="!s1!" goto :no
)
echo yes
goto :eof
:no
echo no

```

WOLGA - January 22, 2016, 5:17 am

I just confirmed both GL1TCH3D and White Light code as working. Very nice job, both of you! ?

Even though White Light's code does not not count spaces, its a trivial fix.

jahwi - January 22, 2016, 3:32 pm

Not extensively tested but this should work:

```

Code @echo off
setlocal enabledelayedexpansion
set one=%1
set two=%2
set one=!one: =!
set two=!two: =!
set answer=yes
set len1=
set len2=
set string=
set stringb=
set found=
::check length of both strings
call :findlength !one!
set len1=!length!
call :findlength !two!
set len2=!length!
if not "!len1!"=="!len2!" set answer=no
for /l %%a in (1,1,!len1!) do (
    set found=no
    set string=!one:~%%a,1!
    for /l %%b in (1,1,!len2!) do (
        set stringb=!two:~%%b,1!
        if "!string!"=="!stringb!" set found=yes
    )
)
if "!found!"=="no" set answer=no
echo !answer!
exit /b
:findlength
echo %1>%temp%\string.tmp
for %%? in (%temp%\string.tmp) do (
    set /a length=%%~z? -2
)
exit /b

```

wurd - January 22, 2016, 5:08 pm

(Jan 22, 2016 01:41 AM)White Light Wrote: [\[CITE\]](#) set s1=%~1

set s2=%~2

set #=%s1%

set l=0

for %%A in (64 32 16 8 4 2 1) do if not "!#:~%%A!"==" set #=!#:~%%A!&set /a l+=%%A! saw this length algo used before but never analyzed it. Just looked at it now and the idea of using binary to find the length is a really clever idea.

Quote:if not "!s2:~%l%!"=="!s2:~%l%,1!" goto :no I was wondering what the purpose of this line is? I understand what it does, I just dont understand why you need it? It just checks to see if the last character is equal to the last character, from my perspective.

White Light - January 22, 2016, 5:54 pm

(Jan 22, 2016 06:17 AM)Qwertyuiopasdfghjklzxc Wrote: [\[CITE\]](#) I just confirmed both GL1TCH3D and White Light code as working. Very nice job, both of you! ?

Even though White Light's code does not not count spaces, its a trivial fix.

Don't know what you're talking about...? There's no reason it won't behave correctly with spaces, and in fact it actually behaves just fine even when you're using all spaces or interspersed.

(Jan 22, 2016 06:08 PM)wurd Wrote: [\[CITE\]](#) I was wondering what the purpose of this line is? I understand what it does, I just dont understand why you need it? It just checks to see if the last character is equal to the last character, from my perspective.

Not exactly, but close. With substrings, the length parameter is optional. Without it, you get from that position to the end of the string. In this case, if the lengths are equal, the position to the end of the string should be exactly one character (the last one), otherwise there will be at least one other character.

Just to reiterate: Say the strings are ABCD and 123456. I know the length of the first string (4, though it's saved as 3 for use), so I can check the second string using that info.

!s2:~3! would be 456 and !s2:~3,1! would just be 4. Since they're different, I know that the two strings are different length. This is a much faster way of comparing lengths than actually checking both (even using an optimized method like 2^n seeks).

7_R3X - January 23, 2016, 3:08 am

I'm not good at scripting.May I give it a try with C++ ? I'm good at it. ?

WOLGA - January 23, 2016, 8:15 am

(Jan 22, 2016 06:54 PM)White Light Wrote: [\[CITE\]](#) Don't know what you're talking about...? There's no reason it won't behave correctly with spaces, and in fact it actually behaves just fine even when you're using all spaces or interspersed.

Quote:Spaces and quotes should not count for checking.

White Light - January 23, 2016, 1:51 pm

But that's stupid. Like, actually really freaking dumb. If you don't want it to count for spaces, just toss in a couple sX=%sX:=% at the start, but that's honestly a really frigging stupid restriction to put on code, even more so than annikk's "sensible" password generator needing to keep a wordlist in the script.

jahwi - January 23, 2016, 3:23 pm

(Jan 23, 2016 02:51 PM)White Light Wrote: [\[CITE\]](#) But that's stupid. Like, actually really freaking

dumb. If you don't want it to count for spaces, just toss in a couple `sX=%sX:=%` at the start, but that's honestly a really frigging stupid restriction to put on code, even more so than annikk's "sensible" password generator needing to keep a wordlist in the script.¹ I did do that in my code

1. Annikk's challenge was fun.
2. `exit /b`

White Light - January 23, 2016, 5:15 pm

(Jan 23, 2016 04:23 PM)jahwi Wrote: [\[CITE\]](#) 1. I did do that in my code

2. Annikk's challenge was fun.

3. `exit /b`

1. Doing it is not stupid. The requirement is stupid and does more harm than good.

2. It was, but he also added in stupid restrictions that do more harm than good.

3. `goto :eof`

jahwi - January 23, 2016, 5:39 pm

(Jan 23, 2016 06:15 PM)White Light Wrote: [\[CITE\]](#) 1. Doing it is not stupid. The requirement is stupid and does more harm than good.

2. It was, but he also added in stupid restrictions that do more harm than good.

3. `goto :eof`format C: /Q

AnHero - January 25, 2016, 2:52 am

(Jan 23, 2016 06:15 PM)White Light Wrote: [\[CITE\]](#) 1. Doing it is not stupid. The requirement is stupid and does more harm than good.

2. It was, but he also added in stupid restrictions that do more harm than good.

3. `goto :eof`

Wait, are you saying that the person who posted the challenge added stupid restrictions? I'm trying to follow. I hope I'm wrong and if so then disregard: I suck cocks.

Otherwise calling whoever crafts the challenge's restrictions stupid is it's stupid. You see insane restrictions put in place on code all the time for legacy, business, or "stupid" reasons.

Working around restrictions is part of the game.

White Light - January 25, 2016, 3:35 am

(Jan 25, 2016 03:52 AM)AnHero Wrote: [\[CITE\]](#) Wait, are you saying that the person who posted the challenge added stupid restrictions? I'm trying to follow. I hope I'm wrong and if so then disregard: I suck cocks.

Otherwise calling whoever crafts the challenge's restrictions stupid is it's stupid. You see insane restrictions put in place on code all the time for legacy, business, or "stupid" reasons.

Working around restrictions is part of the game.

There are restrictions put in place for good reasons, such as legacy purposes and backwards compatibility. Then there are restrictions put in place for no reason other than to add more words to a page. Working around this restriction is no "game", it's at best an annoyance and pedantic. Functionally, there is no reason why you'd include it.