pa08b

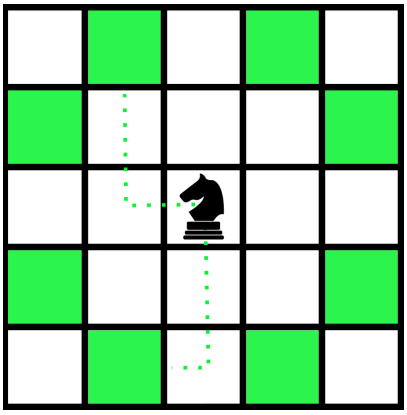
Publish | Edit | ⋮

```
REVISION HISTORY:
2022-11-19: Original
```

pa08b

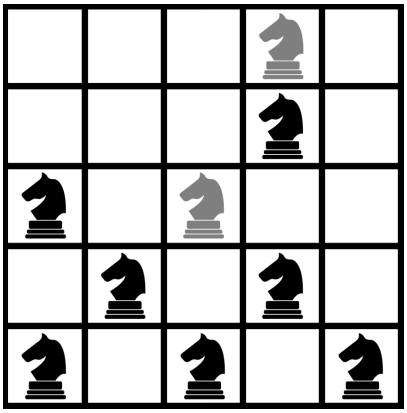
Problem description

In the game of chess, knights are unique due to their "L-shaped" movement. A knight can move, as shown in the figure below, by either moving two squares sideways and one square up or down, or moving one square sideways and two squares either up or down.

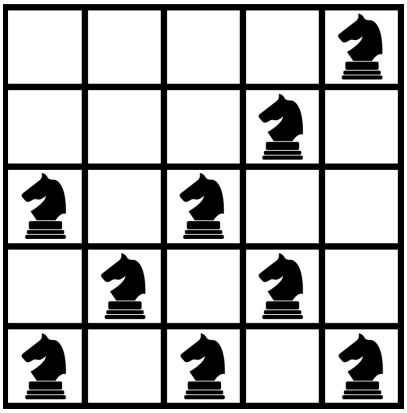


The highlighted squares show all possible moves for a knight.

Exactly nine knights must be positioned on a 5-by-5 board so that no knight can attack another knight with a single move. The configuration shown in the next figure is an invalid solution because two of the knights can attack each other, where the configuration shown in the last figure is a valid solution.



Invalid game configuration.



Valid game configuration.

Given the description of a game configuration, your job is to determine whether or not it represents a valid solution.

Input specification

The input will consist of 5 lines, each having 5 characters. All characters will be either `'K'`, indicating the placement of a knight, or `'.'`, indicating an empty space on the board.

Output specification

If the given chess board is a valid solution to the puzzle, display "0 invalid". Otherwise, display "1 valid".

Sample Input 1

```
...K.
```

```
...K.
K.K..
.K.K.
K.K.K
```

Sample Output 1

```
0 invalid
```

Sample Input 2

```
.....
...K.
K.K.K
.K.K.
K.K.K
```

Sample Output 2

```
1 valid
```

Sample Input 3

```
.....
...K.
K.K.K
.K.K.
K...K
```

Sample Output 3

```
0 invalid
```

Sample interaction

```
$ ./pa08b < pa08b-input0.txt
```

Specific program requirements

- Use only language features introduced in chapters 1-8, lecture, or lab.
- Use named constant(s) to allocate your array(s); do not attempt to use a variable (it will not compile).
- The `main()` function body shall not exceed 25 lines of code.

> Hint: *During development* it's frequently handy to print out an array for visual inspection. Don't

> work in the dark.

Check script

You will find a self-check script (`/home/shared/cs135/kmess/pa08b/pa08b-check`) to diagnose and test your program (Available Thursday.) Until then, remember to check your code with cpplint, cppcheck, and/or pclint tools (in addition to using $CXXFLAGS when compiling at the command line).

> Static analysis tools do not understand our programs. They can only detect things that are demonstrably wrong or are otherwise suspicious. It's up to the developer (us!) to determine if a particular diagnostic applies or not. Strive for as few static analysis diagnostics as possible. It's a good thing.

However,

> Your final program should *compile* cleanly without errors or warnings when using the $CXXFLAGS set of options.

The Judge (TM)

You will find several test inputs and expected outputs in the `/home/shared/cs135/kmess/pa08b/` subdirectory. After copying these to your project directory you can type, for example:

```
$ judge -p ./pa08b -i pa08b-input0.txt -o pa08b-output0.txt
```

Submission

Name your solution **pa08b.cpp** and use the turnin command to submit your solution.

```
$ turnin -c cs135-kmess -p pa08b -v pa08b.cpp
```

Points 100

Submitting Nothing

Due	For	Available from	Until
Nov 28 at 1pm	Everyone	-	-

Programming Rubic						
Criteria	**Ratings**					**Pts**
Program Specifications / Correctness	**40 pts Excellent**	**32 pts Above Average**	**24 pts Average**	**16 pts Below Average**	**0 pts Not Met**	40 pts
Readability	**20 pts Excellent**	**16 pts Above Average**	**12 pts Average**	**8 pts Below Average**	**0 pts Not Met**	20 pts
Documentation	**20 pts Excellent**	**16 pts Above Average**	**12 pts Average**	**8 pts Below Average**	**0 pts Not Met**	20 pts
Code Efficiency	**10 pts Excellent**	**8 pts Above Average**	**6 pts Average**	**4 pts Below Average**	**0 pts Not Met**	10 pts
Miscellaneous	**10 pts Excellent**	**8 pts Above Average**	**6 pts Average**	**4 pts Below Average**	**0 pts Not Met**	10 pts
						Total Points: 100