LEMONEDGE

Development Analysis

Introduction

This allows us to gauge where your strengths lie.

While everything we do is currently in .Net6 (mainly C#), SQL Server and related technologies we are not requiring the answers to be provided in that way. Feel free to use any development tools or languages you are familiar with to complete this test, but .Net is preferred.

This is designed to evaluate your approach, style and understanding of how to solve general programming tasks. It is not a test on your knowledge of .Net, standard libraries, or UI Design ability. In fact, no GUI is required – a console app will suffice.

As such feel free to use your own device, development environment and programming language of choice. Alternatively, we can provide you with a laptop, Visual Studio, and internet access from our offices.

As a guide it should be possible to complete this in around three and a half hours, however you are free to take as much time as you like. During that time feel free to use any online resources and ask myself or colleagues for any assistance or questions you may have.

Submission

When you have finished, please provide all source files, along with a text file indicating the **count for each associated chess piece**. Please include these counts in your response.

We will then be able to evaluate your submission and get back to you with next steps.
Part of which may be a discussion of your solution, involving you walking us through the design choices you made as you approached the problem.

Feel free to reach out to development@lemonedge.com with any other questions you may have.

LEMONEDGE

Problem

The following diagram is of a standard telephone keypad. It consists of a 4x3 grid of buttons. Using the valid moves of a piece from the game of chess, varying combinations of 7-digit phone numbers can be derived. For example, starting in the upper-right corner (the "3" key) using a rook (which moves any number of spaces horizontally or vertically), one valid number, after pressing the initial "3" key, is: 314-5289.

1	2	3
4	5	6
7	8	9
*	0	#

Write a program that will count the number of valid 7-digit phone numbers that can be traced out on the keypad for every given chess piece. The following rules define a valid phone number:

- Seven digits in length
- Cannot start with a 0 or 1
- Cannot contain a * or #

Remember

- This is not a test on your knowledge of development libraries.
- Do not waste time stuck on any one area of functionality, just write a dummy function, and continue.
- Please try to make sure your code compiles and you can talk us through it all when your time is up.
- Please note some of this is deliberately vague to reflect real world scenarios and encourage you to seek answers from myself or the team for clarity on what is required.
- Please feel free to ask as many questions as you like throughout the test as needed.
- Note that it is possible that some pieces may not have any valid phone numbers.

What we are looking for?

- The correct answers for **each individual** chess piece.
- Your solution is maintainable by someone who has not seen it.
- Object-oriented design concepts should be used where-ever they make sense.
- It should be easy to extend the program for new requirements;
    - New keyboard layouts
    - Different rules
    - New types of chess pieces

Questions?

- How do you know what you have developed is correct?
- How can you make it more efficient?