# Coding Evaluation

NOTE: This document should be accompanied by four sample input files.  If you did not receive them, please ask for them before proceeding with the questions. In addition to your answers, we would like to see your code. Please produce your code as if you were producing them for production.

Please produce code in c#

## Question 1

A worker is trying to find the location of the meter for his service order in a large apartment building, but he can't find the right floor - the directions his mobile app are a little confusing. He starts on the ground floor (floor 0) and then follows the instructions one character at a time.

An opening parenthesis, (, means he should go up one floor, and a closing parenthesis, ), means he should go down one floor.

The apartment building is very tall, and the basement is very deep; he will never find the top or bottom floors.

For example:

* (()) and ()() both result in floor 0.
* ((( and (()(()( both result in floor 3.
* ))((((( also results in floor 3.
* ()) and ))( both result in floor -1 (the first basement level).
* ))) and )())()) both result in floor -3.

**To what floor do the instructions take the worker**?

## 

## Question 2

A worker is reading meters in an un-conventional manner in an infinite two-dimensional grid of houses.

They begin by reading a meter at the house at their starting location, and then a dispatcher at meter shop calls them and tells theme where to move next. Moves are always exactly one house to the north (^), south (v), east (>), or west (<). After each move, they read another meter at the house at their new location.

However, the dispatcher at the meter shop is having a bad day and is distracted, and so his directions are a little off, and the meter reader ends up visiting some houses more than once. **How many houses have their meters read at least once**?

For example:

* ">" reads meters at 2 houses: one at the starting location, and one to the east.
* "^>v<" reads meters at 4 houses in a square, including twice to the house at his starting/ending location.
* "^v^v^v" is not very productive and meters are read at only 2 houses.

## 

## Question 3

An implementations team member needs help figuring out which strings in his text file for import are good or bad.

A good string is one with all of the following properties:

It contains at least three vowels (aeiou only), like aei, xazegov, or aeiouaeiouaeiou. It contains at least one letter that appears twice in a row, like xx, abcdde (dd), or aabbccdd (aa, bb, cc, or dd). It does not contain the strings ab, cd, pq, or xy, even if they are part of one of the other requirements. For example:

* ugknbfddgicrmopn is good because it has at least three vowels (u...i...o...), a double letter (...dd...), and none of the disallowed substrings.
* aaa is good because it has at least three vowels and a double letter, even though the letters used by different rules overlap.
* jchzalrnumimnmhp is bad because it has no double letter.
* haegwjzuvuyypxyu is bad because it contains the string xy.
* dvszwmarrgswjxmb is bad because it contains only one vowel.

**How many strings are good**?

## 

## Question 4

The AVL team in client services is trying to wire up a new vehicle modem for testing. Unfortunately assembly is required using a set of wires and bitwise logic gates and they need some help from the development team.

Each wire has an identifier (some lowercase letters) and can carry a 16-bit signal (a number from 0 to 65535). A signal is provided to each wire by a gate, another wire, or some specific value. Each wire can only get a signal from one source, but can provide its signal to multiple destinations. A gate provides no signal until all of its inputs have a signal.

The included instructions booklet describes how to connect the parts together: x AND y -> z means to connect wires x and y to an AND gate, and then connect its output to wire z.

For example:

* 123 -> x means that the signal 123 is provided to wire x.
* x AND y -> z means that the bitwise AND of wire x and wire y is provided to wire z.
* p LSHIFT 2 -> q means that the value from wire p is left-shifted by 2 and then provided to wire q.
* NOT e -> f means that the bitwise complement of the value from wire e is provided to wire f.

Other possible gates include OR (bitwise OR) and RSHIFT (right-shift). If, for some reason, you'd like to emulate the circuit instead, almost all programming languages (for example, C, JavaScript, or Python) provide operators for these gates.

For example, here is a simple circuit:

123 -> x  
456 -> y  
x AND y -> d  
x OR y -> e  
x LSHIFT 2 -> f  
y RSHIFT 2 -> g  
NOT x -> h  
NOT y -> i

After it is run, these are the signals on the wires:

d: 72  
e: 507  
f: 492  
g: 114  
h: 65412  
i: 65079  
x: 123  
y: 456

**Using the instructions (provided as your question input), what signal is ultimately provided to wire ‘a’**?