GEOTAB AFRICA
management by measurement

Junior Software Engineer Assessment

www.geotabafrica.com

Table of Contents

Question 1 - Characters

You are tasked with creating a simple role-playing game in C#. The game involves two types of characters: *Warrior* and *Mage*. Each character has a name, health points (*HP*), and an attack method. The *Warrior* has a special ability called *Slash*, and the *Mage* has a special ability called *Fireball*.

Requirements

1. Implement a *Character* class with properties *Name* and *Health*.
2. Implement methods *Attack* and *TakeDamage* in the *Character* class.
3. Create a *Warrior* and *Mage* class that derive from the *Character* class.
4. Implement special abilities *Slash* and *Fireball* for *Warrior* and *Mage* respectively.
5. Write a main program which simulates a battle between a *Warrior* and *Mage* instance. All available methods must be used.
6. When an action is completed (Attack, Slash, or Fireball), the action must be printed to the console as follows:

   'A' takes 'B' damage, 'C' HP remaining

   Where:

   - A is the name of the character receiving the damage.
   - B is the quantity of damage received by A.
   - C are the remaining health points for character A.

   e.g. Mage takes 10 damage, 90 HP remaining.

Constraints

- Each attack reduces health points by 10.
- The Slash ability reduces health points by 15.
- The Fireball ability reduces health points (HP) by 20.
- The health points cannot go below 0.
- Each character starts out with 50 health points.

Question 2 – Longest Substring Without Repeating Characters

Given a string s, find the longest substring without repeating characters.

Requirements

1. Complete the method *LengthOfLongestSubstring(string s)* that returns the length of the longest substring without repeating characters.
2. Bonus - optimize the solution to run in linear time, O(n).
3. Bonus – write unit tests to test your solution.

Constraints

- 0 <= s.Length <= 5*10^4
- s consists of English letters, digits, symbols, and spaces.

Example

1. Input: “abcabcbb”
   Output: 3
2. Input: “bbbbb”
   Output: 1
3. Input: “pwwkew”
   Output: 3

Question 3 – Word Ladder

Given two words (*startWord* and *endWord*), and a list of words, find the length of the shortest transformation sequence from *startWord* to *endWord*, such that:

1. Only one letter can be changed at a time.
2. Each transformed word must exist in the word list.

Note that *startWord* is not a part of the word list.

Requirements

1. Implement a method *LadderLength(string startWord, string endWord, IList<string> wordList)* that returns the length of the shortest transformation sequence from *startWord* to *endWord*, or 0 if no such sequence exists.
2. Bonus – Optimize the solution for performance.
3. Bonus – Write unit tests to test your solution.

Constraints

- All words have the same length.
- All words contain only lowercase alphabetic characters.
- The length of *wordList* will be between 1 and 5000.
- The length of each word will be between 1 and 10.

Example

- Input:
    - startWord: “hit”
    - endWord: “cog”
    - wordList: [“hot”, “dot”, “dog”, “lot”, “log”, “cog”]
- Output: 5

  The shortest transformation sequence is “hit” → “hot” → “dot” → “dog” → “cog”, which is 5 steps.

Example

- Input:
    - startWord: “hit”
    - endWord: “cog”
    - wordList: [“hot”, “dot”, “dog”, “lot”, “log”]
- Output: 0

  The endWord “cog” is not in the wordList, so no transformation sequence exists.