```
import unittest
from ruzzle import *
from functools import *

known_inputs = [
  ["walk", "moon", "hate", "rope"],
  ["reca", "rwar", "aazp", "syon"],
  ["abse", "imtn", "nded", "ssen"]
 ]

offsets = [( 1, 0),( 0, 1),(-1, 0),( 0,-1)]

def check(word, m, i, j):
  tmp = False
  if word == '': return True
  if ( (0 <= i < 4) and (0 <= j < 4) ) and (m[i][j] == word[0]):
    for offsetx, offsety in offsets:
      tmp |= check(word[1:], m, i+offsetx, j+offsety)
  return tmp

def is_hidden(word, grid):
  m = [list(row) for row in grid]
  for i, j in [(x,y) for x in range(4) for y in range(4)]:
    if check(word, m, i, j): return True
  return False

def is_duplicated(w, grid):
  g = reduce(lambda x,y: x+y, grid)
  wl = { c : w.count(c) for c in w }
  gl = { c : g.count(c) for c in g }
  return all( wl[c] <= gl[c] for c in wl )

class RuzzleTest(unittest.TestCase):
  def setUp(self):
    self.words = [w for w in open('wl.txt').read().split() if (3<=len(w)<=16)]
    self.known_values = [(inp, ruzzles(inp)) for inp in known_inputs]
  def test_sorting(self):
    for inp, outp in self.known_values:
      self.assertEqual(outp,sorted(outp))
  def test_length(self):
    for inp, outp in self.known_values:
      self.assertTrue(all(3 <= len(elem) <= 16 for elem in outp))
  def test_english(self):
    for inp, outp in self.known_values:
      self.assertTrue(all(elem in self.words for elem in outp))
  def test_twice(self):
    for inp, outp in self.known_values:
      self.assertTrue(all(is_duplicated(elem, inp) for elem in outp))
  def test_hidden(self):
    for inp, outp in self.known_values:
      self.assertTrue(all(is_hidden(elem, inp) for elem in outp))

if __name__ == "__main__":
  unittest.main()
```