# 问题一

小明住在未来市，这个地方所有的道路都切割为完美的网格状。有一天他与一位朋友有约，但他提前了十分钟抵达约定地点，所以他决定先去散散步。这个城市为市民提供了一个Random Walk App，每次按下执行便会发送一串代表行走方向且长度不定的单字字母字符串（例如[‘n’, ’s’, ’w’, ’e’]及代表依序往北、南、西、东。小明只有十分钟，每次一个方向就走一个街口刚好需要一分钟，若小明需要在十分钟后准时回到原点（不可过早或过晚），请为小明设计一个方法名为walkValidator，回传为true时代表Random Walk App生成的路线可以达到这个要求，回传false则代表生成的路线无法达成。

function walkValidator(walk) {

*//在此输入你的方法内容*

let direction = {

n: 1,

s: -1,

e: 1,

w: -1

}

return walk.**reduce**((*pre*, *cur*) => pre + direction[cur], 0) === 0

}

//一些测试可能如下...

Test.expect(walkValidator (['n','s','n','s','n','s','n','s','n','s']), ‘true');

Test.expect(!walkValidator (['w','e','w','e','w','e','w','e','w','e','e','w']), ‘false');

Test.expect(!walkValidator (['e']), ‘false');

Test.expect(!walkValidator (['n','n','n','s','n','e','e','s','s','s']), ‘false');

# 问题二

小东是个每天走路上下班的上班族，早上出门和晚上回家时会看天气预报，如果早上出门时下雨，则需要从家里带一把伞出门，并在抵达公司后将伞放在公司。同样地，如果下班时下雨且公司有伞可带回，则他会带伞回家，如果没有伞，则必须要去买一只伞回去以便他永远不会被淋湿。但若没有下雨，则小东永远不会拿伞。

若在给定一连串的天气条件下，你的任务是必须为小东设计出他所需要的最少数量雨伞方法amoutOfUmbrellas。例如[‘下雨’,’晴天’,’下雨’,’下雨’,’晴天’,’晴天’]，则代表三天的预报，依序为第一天早上下雨下班晴天，第二天早上下雨下班下雨，依此类推。

function amoutOfUmbrellas(weather) {

*//在此输入你的方法内容*

if (weather.length === 1) return 0;

let umbrellas = 0;

let home = 0, company = 0;

for (let i = 0; i < weather.length; i++){

let w=weather[i]

if (i % 2 === 0) {

if (w === '下雨') {

if (home === 0) {

umbrellas++;

company++;

} else {

home--;

company++;

}

}

} else {

if (w === '下雨') {

if (company === 0) {

umbrellas++;

home++;

} else {

company--;

home++;

}

}

}

}

return umbrellas;

}

//一些测试可能如下...

Test.assertEquals(AmoutOfUmbrellas ([‘下雨’]), 0, ‘只有一个时段的预报不计算’);

Test.assertEquals(AmoutOfUmbrellas ([‘下雨', ‘下雨‘, ‘下雨‘, ‘下雨‘]), 1, ‘始终下雨应为一把伞’);

Test.assertEquals(AmoutOfUmbrellas ([‘晴天’, ‘下雨‘, ‘晴天’, ‘下雨‘,), 2, ‘两个上班晴天，下班雨天需要两把伞’);

# 问题三

请设计一个方法isomorphValidator，此方法能够确认两给定字符串是否为同构（Isomorphic）。

同构的定义在此为：两个字符串A、B中每个字符都能互相映像，则字符串A和B可称为同构。

例如字符串A为’ABB’，字符串B为’CDD’。则字符串A的A可映像到字符串B的C，字符串A的B可映像道字符串B的D，彼此存在互向映像关系，所以同构。

字符串C为’ABAB’，字符串D为’CD’，因为字符串C的第三和第四字符在字符串D中没有字符可映像，所以字符串C和D非同构。

function isomorphValidator (a,b) {

*//在此输入你的方法内容*

if (a.length !== b.length) {

return false;

}

let index = 0;

let hash = {};

for (let i = 0; i < a.length; i++) {

let t = a[i];

if (!hash[t]) {

hash[t] = b[i];

index++

} else {

if (hash[t] !== b[i]) {

return false;

}

}

}

return a.length !== index && true;

}

//一些测试可能如下...

Test.assertEquals(isomorphValidator ("ESKAKE", "GUELEG"), true)

Test.assertEquals(isomorphValidator ("GGG", "YYY"), true)

Test.assertEquals(isomorphValidator ("SUN", "SON"), false)

Test.assertEquals(isomorphValidator ("AAB", "BAA"), false)

// Should handle words with more than 10 characters

Test.assertEquals(isomorphValidator ("aslfmvfkie","apelfmfjrw"), false);