1.1.先 mock 的作用就是框定物件，讓它不會再改變，因為 mock object 的概念是創造一個物件來跑看看結果是否如預期，過程中會預設一些行為和 return 的值 ，最後來看是否如我們預想的被調用，來查看測試中的互動是否正確。

所以在 require('./main') 之前 mock fs.readFile，不會真正的fs.readFile，確保我們有掌控權，這可以用來測試，也就是讓變項只有一個。

1.2.可不可以寄送郵件的部分，原先的 code 裡面 send 是依賴 random 的結果來控制發送成功與否，而 random 是隨機數，沒有辦法確定會是什麼樣的結果，所以先 mock 的話控制結果才能測試，就會跟1.一樣的概念。

1.3. 先說明對於你在帶LAB2的時候，5 種 test double 的理解：

- Test Stub：

- 看是否正確調用

- 用於隔離測試，依賴呼叫外部的模擬行為

- Mock Object

- 創造一個物件實際跑一次看結果是否符合預期

- 設定預期的行為 & return 的值，然後在測試結束時驗證是否如預期般被正確調用，用於檢查互動是否正確。

- Test Spy

- 類似更詳細版本的 Test Stub、Mock Object，記錄了調用的次數、參數等等

- 專注在「副作用」，譬如調用某個方法後，除了驗證 return 的結果以外，是否會影響其他系統，譬如全域變數

- Fake Object

- 跟Test stub 類似，不過Fake Object 是創造一個假的介面做測試

- Test Tsub 則是用真的介面， return 一個假的值

- Dummy Object

- 不參與邏輯測試，就是塞滿語法需求來做測試

從前面我的理解app.mailSystem.send = test.mock.fn(app.mailSystem.send)後assert app.mailSystem.send.mock.calls.length 的方式屬於Test Spy，因為用 Spy 可以在不改變 send 方法下來看是否正確的調用，看看調用的次數跟預期有沒有落差。

1.4. Ban -> jack -> Peter

 ['Peter', 'jack', 'Ban']

索引從0開始，length - 1 就是2，也就是 Ban，cnt-- 也就是每一次 -1 ，所以就是Ban -> jack -> Peter

1.5.如同前面1.2.3.說的，mock object 來確保random 預設的值，就不用對原本的 code 修改。

2.1.

createMockFn，跟1.的1.2.3 提到mock object一樣的概念，使用原本的邏輯和預設值，來確保結果有沒有跟預期的一樣。

在測試前後設置setup暫存檔和teardown清理環境。這是Test Stub用來隔離測得的時候用的，隔離可以讓每次測試都是在乾淨的環境，避免互相干擾，也跟做ARP攻擊的模擬時雷同，像是有時候要清理browser的cache，避免操作過程被cache汙染，這樣測試結果才會正常。

預設的MailSystem.write 是字串，所以檢查input的type是否正確，有點類似recursive 對於 boundary condition都會特別注意的概念，因為追蹤各種type的結果，所以是test spy 。

Main.js裡面const success = Math.random() > 0.5;

MailSystem.send這邊可以設定值0.9 & 0.1，來測試比0.5大or小的狀況，這樣就可以測到成功和失敗兩種情形。

Application的init和def功能有沒有正常運作，看assert.strictEqual(app.people.length, 3, 'Should load 3 people')

有沒有正確地把名字正確載入

app.mailSystem.write = createMockFn(app.mailSystem.write);

app.mailSystem.send = createMockFn(app.mailSystem.send);

使用 2個Mock object 來看mailSystem.write 和 mailSystem.send 方法的調用情形。

Application.selectNextPerson如果 return null，代表已經沒有人可以選了，也代表把所有的名字都選完。