シモンとアリスは、旗上げゲームで遊んでいます。アリスは赤旗と白旗を持っており、それぞれの旗は上がっているか下がっているかの 22 つの状態を持ちます。はじめ、アリスが持っている両方の旗は下がっています。

シモンは、アリスに

* 「赤を上げて」
* 「赤を下げて」
* 「白を上げて」
* 「白を下げて」

の 44 パターンの指示を q*q* 個出します。アリスは非常に優秀なので、この旗の指示に対して常に正しく行動します。逆にシモンは、以下のような間違った指示を出してしまうことがあります。

* 既に上がっている旗に対して、間違えて「上げて」という指示を出す
* 既に下がっている旗に対して、間違えて「下げて」という指示を出す

このような間違った指示をしてしまうと、シモンはルール違反で敗北し、アリスが勝利します。もし、全司令を通して、間違った指示を出さなければシモンの勝ちです。

あなたは、シモンが出した全てのログを持っています。全ての指示を終えた後、勝つのはシモンでしょうか、アリスでしょうか？もしシモンが勝ったのであれば、最後のアリスの旗の状態を出力してください。

**ザ・ミッション**

指示の記録から、どちらが勝ったかを判定し、シモンが勝ったのであれば最後のアリスの旗の状態を出力するプログラムを作成してください。

**実装の詳細**

**CLI**

入力値が標準入力から渡されて、答えを標準出力に表示するCLIアプリケーションを実装してください。  
詳細は末尾の「CLI アプリ作成用テンプレート」を参照ください。

**入力ルール**

以下のフォーマットに従う標準入力が与えられます。

s[1] s[2] ... s[q]

制約は以下です。

* 1 \le q \le 501≤*q*≤50, 整数
* s[i]*s*[*i*] はRU, RD, WU, WDのいずれか
  + s[i]*s*[*i*] の 11 文字目は、旗の色を表しています。 11 文字目がRの場合は赤(Red)、Wの場合は白(White)の旗に対する指示であることを表しています
  + s[i]*s*[*i*] の 22 文字目は、上げるか下げるかを表しています。 22 文字目がUの場合は上げる指示(Up)、Dの場合は下げる指示(Down)であることを表しています

**出力ルール**

以下のフォーマットを満たす標準出力を出力してください。

Winner

**[FlagStates]**

* 11 行目に、シモンが勝利した場合はSimon、アリスが勝利した場合はAliceと出力してください。
* もしシモンが勝利した場合、 22 文字の旗の状態XYを表示してください。
  + Xには、赤の旗が上がっているならばU、下がっているならばDを記入してください。
  + Yには、白の旗が上がっているならばU、下がっているならばDを記入してください。

**入出力例**

**入出力例 1**

標準入力

RU WU WD

標準出力

Simon

UD

間違った指示はなく、以下のようにゲームが進行します。

* 始め、アリスの旗は赤も白も下がっている。
* 11 個目の指示で、赤が上がる。アリスの旗は赤が上がっており、白は下がっている。
* 22 個目の指示で、白が上がる。アリスの旗は赤も白も上がっている。
* 33 個目の指示で、白が下がる。アリスの旗は赤が上がっており、白は下がっている。

よって、最終的に赤が上がっており、白が下がっている。答えとしては、シモンが勝ち、その時の最終状態はUDであると答えるのが正解となります。

**入出力例 2**

標準入力

RD RU WU RD

標準出力

Alice

11 回目の指示の時点で、シモンが間違った指示を出しているので、アリスの勝利となります。シモンは、敗北したことに気づかず、この例のように更に指示を何個か送ってしまうこともあります。

なお、期待する入出力は [test/basic\_testcases.json](https://app.tracks.run/editor/test/basic_testcases.json) にいくつか定義されています。  
実装する際の参考にしてください。

if \_\_name\_\_ == '\_\_main\_\_':

alice = {

"R": "D",

"W": "D"

}

simon = list(input().split())

flag = 0

for inst in simon:

if alice[inst[0]] == inst[1]:

flag = 1

break

else:

alice[inst[0]] = inst[1]

if flag == 0:

print("Simon")

print("{}{}".format(alice["R"], alice["W"]))

else:

print("Alice")

## ザ・ミッション

下記のホストにAPIが実装されています：

<http://challenge-server.code-check.io/>

このホストに実装されている hash APIをコールするCLIアプリケーションを作成してください！

## 実装の詳細

#### API仕様

APIの仕様は以下の通りです。

* Endpoint: /api/hash
* HTTP Method: GET
* Parameters(Query)
  + q : ハッシュを生成する文字列。必須。
* Response:
  + 形式は以下の2つのキーを持つJSONです。
    - q : 与えられたqパラメータ
    - hash : 生成されたハッシュ値
  + APIの実行に成功した場合は、200 OKでJSONが返ります。
  + APIの実行に失敗した場合(qパラメータが指定されなかった場合)は、400 Bad Requestが返ります。
  + 同じqパラメータに対しては必ず同じハッシュ値が生成されます。
  + 例:

{"q":"hoge", "hash":"ac0030f68eee4280a8cff568b36dc7e8944880fcab1eadd9220c6c583939baf8"}

#### CLI

入力値を引数に取り、結果を標準出力に出力するCLIアプリケーションとして解答を実装してください。  
詳細は末尾の「CLI アプリ作成用テンプレート」を参照ください。

#### 入力ルール

* 作成するCLIアプリケーションは引数をひとつだけ取ります。
  + 引数はAPIのqパラメータに設定する値です。
  + 制御文字を除くASCII文字からなる文字列であることが保証されます。
  + 文字列の長さは1文字以上100文字以下であることが保証されます。

#### 出力ルール

* 生成されたハッシュ値を標準出力に出力してください。
  + 引数として与えられたqパラメータを用いてAPIを実行してください。
  + APIの実行には任意のライブラリを使用して構いません。

#### 入出力例

**$** ./myApp hoge

ac0030f68eee4280a8cff568b36dc7e8944880fcab1eadd9220c6c583939baf8

なお、期待する入出力は [test/](https://app.tracks.run/editor/test/) ディレクトリにいくつか定義されています。  
この問題では入力ファイルの1行目全体がそのまま、ひとつの引数としてアプリケーションに渡されます。  
空白やダブルクオートなどが入力ファイルに含まれる場合、アプリケーションが受け取る引数の値は入力ファイルの内容そのままです。  
実装する際の参考にしてください。

# コマンドラインアプリケーション(CLI アプリ)作成用テンプレート(Python3.x)

[main.py](https://app.tracks.run/editor/main.py)を編集して、CLIアプリを実装してください。  
チャレンジ内でファイルの作成が許可されていれば、可読性等のためにファイルを分割する事も可能です。

## コマンドライン引数の取得方法

[main.py](https://app.tracks.run/editor/main.py)内で定義されている、mainという関数から、 argv の名前で取得可能です。

**def** **main**(argv):

*# code to run*

ここでの argv は sys.argv の内容からスクリプト名の情報を抜いたデータが入ります。

## 入力データファイルの取得方法

入力データがファイルの場合、open関数を使用してください。

f = open(v)

line = f.readline()

**while** line:

print(line.replace('\n',''))

line = f.readline()

f.close()

## コマンド実行結果の標準出力への出力

stdoutへの出力は標準のprintメソッドが使用可能です。

print(result)

import requests

import sys

if \_\_name\_\_ == '\_\_main\_\_':

  q = sys.argv[1]

  param = {"q": q}

  url = "http://challenge-server.code-check.io"

  endpoint = "/api/hash"

  response = requests.get(url+endpoint, params=param)

  print(response.json()["hash"])