Szablon rozwiązania	`egzP6a.py`
Złożoność akceptowalna (1.5pkt):	O(*nk* + *n*log*n*)
Złożoność wzorcowa (+2.5pkt):	O(*nk*), gdzie ***n*** to liczba haseł, a ***k*** to max. długość hasła

W tym zadaniu nie wolno korzystać z wbudowanych funkcji sortowania.

W ciągu ostatniej dekady odbyło się bardzo dużo włamań na konta użytkowników w wielu różnych serwisach internetowych. Google postanowiło pomóc w rozwiązaniu tego problemu poprzez stworzenie wytycznych na temat siły haseł, które powinny zostać ustawiane przez użytkowników. Do tej pory weryfikacja siły hasła odbywała się słabszym, starym systemem, jednak w wyniku konsultacji, został opracowany nowy system, oparty na danych statystycznych. Korzystając z danych zdobytych w wyniku wielu wycieków, Google posiada listę różnych haseł, ustawianych w przeszłości przez użytkowników różnych serwisów. Postanowiono, że zostaną one posortowane według siły hasła, a następnie zostanie wybrana pewna ilość najlepszych haseł, które będą akceptowane jako silne. Jako, że nowy system opiera się na statystyce a nie twardych kryteriach, potrzebna jest informacja jakie jest możliwie najsłabsze hasło, które zostanie dopuszczone przez system. Jeżeli będzie kilka tak samo słabych, wystarczy podać dowolne z nich.

Zadanie polega na zaimplementowaniu funkcji:

```
google( H, s )
```

która wskaże najsłabsze dopuszczalne przez system hasło, przy następujących założeniach:

1. Tablica **H** zawiera wszystkie hasła posiadane przez Google wyrażone jako ciągi znaków, zmienna **s** określa ilość haseł, która będzie akceptowana, jako silne.

2. Rozważając siłę dwóch haseł, rozważane są (w kolejności) następujące kryteria:
   - Silniejsze jest hasło, które jest dłuższe (ma większą łączną liczbę znaków)
   - W przypadku takiej samej ilości znaków, silniejsze jest to, w którym jest więcej liter

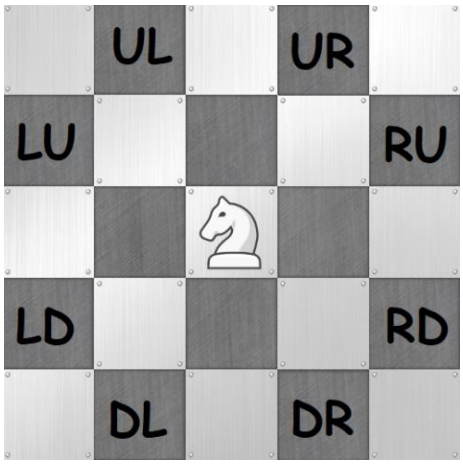
3. Hasła będą składać się jedynie z małych liter alfabetu angielskiego oraz cyfr od 0 do 9, aczkolwiek na cele oszacowania złożoności obliczeniowej należy przyjąć, że liczba dopuszczalnych znaków nie jest stała (w szczególności mogłaby być ona bardzo duża)

Rozważmy następujące dane:

```
H = ['aba', 'abc', 'ab1', 'abab', 'a1a1', 'aa12a']
s = 3
```

Wywołanie funkcji `google( H, s )` powinno zwrócić wynik **a1a1** (Kolejność haseł według siły to **aa12a** -> **abab** -> **a1a1** -> **aba** -> **abc** -> **ab1**)

Szablon rozwiązania	`egzP6b.py`
Złożoność akceptowalna (1.5pkt):	O(*n*log*n*)
Złożoność wzorcowa (+2.5pkt):	O(*n*), gdzie ***n*** to liczba skoków Garka



W magicznej krainie istnieje nieskończona szachownica, w której pola określna się przez współrzędne całkowite (również ujemne). Dodatkowo każde pole posiada żarówkę, która zmienia swój stan od razu, kiedy jakaś figura szachowa robi ruch na to pole. Początkowo każde pole na szachownicy było zgaszone, gdy nagle Konik Szachowy Garek został zrzucony z odrzutowca na pole o współrzędnych (0, 0) – oczywiście, w wyniku zrzucenia żarówka na polu (0, 0) zapaliła się. Garek zdezorientowany tą sytuacją zaczął wykonywać losowe ruchy dokładnie w taki sposób, jak na skoczka przystało. Skoczek może wykonać każdy z ośmiu skoków pokazany na rysunku z równym prawdopodobieństwem.

Zadanie polega na zaimplementowaniu funkcji:

```
jump( M )
```

która zwraca liczbę zapalonych żarówek, po wykonaniu wszystkich ruchów znajdujących się w tablicy **M**. Tablica **M** zawiera listę oznaczeń tak jak na powyższym rysunku, a i-ty element w tej tablicy odpowiada za (i+1)-ty ruch Garka.

Rozważmy następujące dane:

```
M = ['UL', 'RD', 'LU', 'LU', 'RD', 'DL', 'UR', 'DR']
    # ↰   ↳   ↰   ↰   ↴   ↲   ↱   ↳
```

Wywołanie funkcji `jump( M )` powinno zwrócić wynik **3.** Po przejściu przez te pola, Konik Szachowy Garek będzie znajdował się na polu o współrzędnych (0, 0), natomiast żarówki będą się świeciły na polach **(1, 1)**, **(-2, 0)** oraz **(-3, 3)**

Złożoność obliczeniowa. W trakcie rozwiązywania tego zadania należy przyjąć, że złożoność funkcji zapisu/odczytu ze słownika w Pythonie wynosi O(1)