3. domácí úkol

- zadaný: 23. 10. 2023
- deadline: 6. 11. 2023, **23:55**

Pokyny

- Odevzdejte jediný soubor pojmenovaný prijmeni_du3.py (“prijmeni” nahraďte svým příjmením) s definicemi požadovaných funkcí. Odevzdávárnu najdete v ISu: Student / FI:IB113 / Odevzdávárny / Skupina 10 / DU3
- Šablona řešení je dostupná ve studijních materiálech.
- Jména odevzdávaných funkcí neměňte; ať zůstanou pojmenované stejně, jako jsou pojmenované v zadání.
- Případné dotazy pokládejte v diskuzním fóru této skupiny ve vláknu k tomuto úkolu.
- Příklad vypracujte samostatně. Nepovolená spolupráce bude potrestána dle pravidel předmětu.
- Funkce, které napíšete, můžete používat v rámci jiných funkcí. Můžete si implementovat i další pomocné funkce.

Zadání

Vytvořte simulátor zjednodušené hry „Hadi a žebříky“.
Pravidla:

- Hraje se na hracím plánu o *n* polích.
- Hraje pouze jedna figurka, která začíná na jedné straně hracího plánu, na druhé straně je domeček, kam se chce dostat.
- Háže se kostkou (1-6), když padne 6, hází se znovu (až nekonečněkrát).
- Figurka se posunuje buď o celkový součet hodů na kostkách, nebo vůbec (v situacích, kdy by přešla domeček).
- Každé *k*-té pole obsahuje hada. Když na něj figurka vstoupí, vrací se na začátek.
- Hra končí, když figurka dorazí přesně na poslední pole. Pokud by na posledním poli měl být had, figurka se už nevrací na začátek a hra končí i tak.

Poznámky:

- Hrací pole číslujeme od 1.
- Můžou se vám hodit funkce pro generování náhodných čísel, funkce random.seed (pro generování náhodných sekvencí stejných při každém spuštění), cyklus for i cyklus while, celočíselné dělení // i zbytek po dělení %.
- V tomto domácím úkolu nebudete potřebovat seznamy, ani práci s řetězci složitější než jejich výpis.

Vaším úkolem bude naprogramovat následující funkce:

Házení kostkou

Napište funkci `get_throw_total()`, která **vrátí** náhodný počet políček, o které se má figurka posunout důsledkem hodu, tj. při hodu 1-5 vrátí hozenou hodnotu, při hodu 6 součet této hodnoty a dalšího hodu.

Každý z dílčích hodů se realizuje férovou šestistranou kostkou (všechna čísla mají stejnou pravděpodobnost).

```
>>> get_throw_total()
3
>>> get_throw_total()   # tady jsme hodili 6 a pak 4
10
>>> get_throw_total()
1
```

Výpis herního pole

Napište funkci `print_board(position, length, k)`, která **vypíše** herní plán. Prázdná políčka nechť jsou značena `.`, políčka s hadem `s` a políčko s figurkou `f`. Zároveň vypište i pozice políček dle příkladu níže.

```
>>> print_board(6, 42, 5)
....sf...s....s....s....s....s....s....s..
0000000001111111111222222222233333333334444
1234567890123456789012345678901234567890123

>>> print_board(16, 16, 4)
...s...s...s...f
0000000001111111
1234567890123456
```

Simulace tahu

Napište funkci `turn(position, length, k, output)`, která provede jeden tah ve hře podle pravidel popsaných na začátku zadání a **vrátí** pozici figurky na konci tahu. Počáteční pozice figurky je `position`, délka herního plánu je `length`, každé `k`-té pole obsahuje hada vracejícího na počátek (viz pravidla). Pokud je parameter `output` True, měla by funkce navíc **vypsat** celkovou hozenou hodnotu (ve tvaru níže) a herní plán ve stavu na konci tahu.

```
>>> turn(6, 42, 5, False)   # hodili jsme 3
9
>>> turn(9, 42, 5, False)   # hodili jsme 1 a kvuli hadovi
1                           # sli na zacatek
```

```
>>> turn(1, 42, 5, True)   # hodili jsme 6 a pak 4 a vypsali stav
Throw total: 10
....s....sf...s....s....s....s....s....s..
00000000011111111112222222222333333333344 4
12345678901234567890123456789012345678901 2
11   # krome toho jsme vratili (nevypsali) soucasnou pozici

>>> turn(11, 42, 5, True)   # hodili jsme 4, kvuli hadovi se vratili
Throw total: 3              # a vypsali stav
f...s....s....s....s....s....s....s....s..
00000000011111111112222222222333333333344 4
12345678901234567890123456789012345678901 2
1   # krome toho jsme vratili (nevypsali) soucasnou pozici
```

Simulace hry

Napište funkci `simulate_game(length, k, output)`, která provede simulaci jedné hry na hrací ploše délky `length` s hadem na každém `k`-tém políčku a **vrátí** počet tahů, kolik hra trvala. Pokud je hodnota parametru `output` True, funkce navíc po každém kole **vypíše** současný stav hry. Na průběh hry se vztahují pravidla uvedená na začátku zadání.

```
>>> simulate_game(20, 5, True)
Throw total: 3
...fs....s....s....s
00000000011111111112
12345678901234567890

Throw total: 4
....s..f.s....s....s
00000000011111111112
12345678901234567890

Throw total: 2
f...s....s....s....s
00000000011111111112
12345678901234567890

Throw total: 3
...fs....s....s....s
00000000011111111112
12345678901234567890
```

```
Throw total: 8
....s....s.f..s....s
00000000011111111112
12345678901234567890

Throw total: 5
....s....s....s.f..s
00000000011111111112
12345678901234567890

Throw total: 4      # figurka se neposunula, presla by cil
....s....s....s.f..s
00000000011111111112
12345678901234567890

Throw total: 2
....s....s....s...fs
00000000011111111112
12345678901234567890

Throw total: 4
....s....s....s...fs
00000000011111111112
12345678901234567890

Throw total: 1
....s....s....s....f
00000000011111111112
12345678901234567890

10   # navratova hodnota (pocet tahu)
```

Analýza her

Napište funkci `analyze_game(length, k, sim_count)`, která provede simulaci `sim_count` her délky `length` s hady na každém `k`-tém poli a **vypíše**, jaká je průměrná délka hry. Funkce by neměla jinak nic průběžně vypisovat.

Formát výpisu průměrné délky by se měl přesně shodovat s příkladem (tj. číslo nemusíte nijak upravovat):

```
>>> analyze_game(11, 5, 100)
Average length: 10.3
```

```
>>> analyze_game(11, 5, 100)
Average length: 8.84
```