Problem 15: LMCoin

Punkty: 35

Autor: Ben Fenton, Faslane, Helensburgh, Wielka Brytania

Wprowadzenie do problemu

W dzisiejszych czasach obserwujemy szybki wzrost popularności bitcoina i innych kryptowalut. Różnią się one znacząco od tradycyjnych środków płatności, bo nie wymagają udziału banku ani instytucji wydającej karty kredytowe jako pośrednika między nabywcą i sprzedawcą. Dzięki temu strony transakcji nie muszą przekazywać prowizji do banku czy innej organizacji.

Zamiast tego bitcoin trafia bezpośrednio do drugiej strony transakcji. Jednak wiąże się to z istotnym problemem - jak dostarczyć dowód zapłaty za towar? Albo jak w ogóle dowieść faktu posiadania pieniędzy, jeśli nie liczyć możliwości uzyskania poświadczenia innej osoby? Jest to znane pod nazwą „problemu podwójnego wydawania".

Zamiast banku, który rejestruje wszystkie transakcje w głównej księdze to użytkownicy bitcoina rejestrują wszystkie transakcje jednocześnie. Oznacza to, że próba oszukania systemu zostanie zauważona, a transakcja nie zostanie przeprowadzona. Służy do tego zabezpieczenie zwane łańcuchem bloków (ang. blockchain). W omawianym problemie zbudujemy prosty łańcuch bloków dla LMCoin, naszej własnej waluty cyfrowej.

Jak sugeruje nazwa, łańcuch bloków zawiera kilka „bloków" danych, z których każdy reprezentuje oddzielną transakcję. Każdy blok jest identyfikowany za pomocą niepowtarzalnego hash'a (funkcja skrótu), wartości generowanej z wykorzystaniem wszystkich informacji przechowywanych w bloku; zalicza się do nich hash poprzedniego bloku w łańcuchu. Dlatego też w miarę przesuwania się wzdłuż łańcucha jego spójność rośnie i gwarantuje, że poprzednie transakcje nie ulegną zmianie. Bitcoin i inne kryptowaluty są tworzone w procesie zwanym „kopaniem"; dodawaniem nowego bloku do łańcucha, który tworzy niepowtarzalny „hash" w danym zakresie - wiąże się to ze sporym wysiłkiem włożonym w zgadywanie!

Opis problemu

Waszym zadaniem jest napisanie programu implementującego łańcuch bloków LMCoin. Jak sugeruje nazwa, łańcuch bloków zawiera kilka „bloków" danych, z których każdy reprezentuje oddzielną transakcję. Oprócz bloku startowego (który nie zawiera elementu poprzedniego bloku) każdy z bloków zawiera cztery informacje:

1. Sygnaturę czasową, przedstawiającą czas wygenerowania bloku
2. Pewne dane (np. zamówienie na pizzę)
3. Indeks (położenie danego bloku w łańcuchu)
4. Hash poprzedniego bloku w łańcuchu

Algorytm hash'a wykorzystywany w naszej walucie działa następująco:

$$H_n = \frac{(T_n + V_n + n + H_{n-1}) * 50}{147}$$

Gdzie:

- n to indeks bloku w łańcuchu; pierwszy blok w łańcuchu ma indeks 1.
- H_n to hash bloku o indeksie n. ($H_0 = 0$)
- T_n to sygnatura czasowa bloku o indeksie n.
- V_n to wartość numeryczna danych zawartych w bloku o indeksie n (wyjaśnienie poniżej).

Dane bloku są zamieniane na wartość numeryczną przez zsumowanie wartości każdej litery w łańcuchu danych, zgodnie z poniższą regułą:

Litera	a	b	c	d	e	f	g	h	i	j	k	l	m
Wartość	1	2	3	4	5	6	7	8	9	10	11	12	13
Litera	n	o	p	q	r	s	t	u	v	w	x	y	z
Wartość	14	15	16	17	18	19	20	21	22	23	24	25	26

Na przykład, wartość łańcucha „cheese" to 3 + 8 + 5 + 5 + 19 + 5 = 45.

Sygnatury czasowe są podawane w formacie DDMMRRHHMM (dwie cyfry dla, kolejno, dnia, miesiąca, roku, godziny i minuty); na przykład, 27 kwietnia 2019 r., 8:30 zostanie zapisana jako 2704190830.

Przykładowe dane wejściowe

Pierwszy wiersz danych wejściowych waszego programu, **otrzymanego przez standardowe wejście**, będzie zawierać dodatnią liczbę całkowitą oznaczającą liczbę przypadków testowych. Każdy przypadek testowy będzie zawierać następujące wiersze danych wejściowych:

- Wiersz zawierający listę wartości danych dla pierwszych dziesięciu bloków w łańcuchu, oddzielonych spacjami. Wartości danych będą zawierać jedynie małe litery.
- Wiersz zawierający listę sygnatur czasowych dla pierwszych dziesięciu bloków w łańcuchu, oddzielonych spacjami. Sygnatury czasowe będą w formacie podanym powyżej.

```
2
pepperoni veggie ham peppers cheese olives mushroom chicken beef bacon
2704191000 2704191030 2704191100 2704191130 2704191200 2704191230
2704191300 2704191330 2704191400 2704191430
candy candy salad chips pretzel icecream apple fries cookie sandwich
2602201200 2602201300 2702201200 2702201300 2802201200 2802201300
2902201200 2902201300 0103201200 0103201300
```

(Należy pamiętać, że po liczbie przypadków testowych znajdują się tylko cztery wiersze danych wejściowych; po prostu wiersz z sygnaturami czasowymi jest za długi i nie mieści się na stronie.)

Przykładowe dane wyjściowe

W każdym przypadku testowym wasz program powinien wyświetlić hash dziesiątego bloku w łańcuchu, obliczony z wykorzystaniem podanych wartości. Wyświetlone hash'e mają być zaokrąglone do najbliższej pełnej liczby. Nie wolno zaokrąglać pośrednich wartości hash'y używanych do obliczeń.

```
1393884230
219309065
```