



Problem 4. «*nsucoin*»

Alice, Bob, Caroline and Daniel are using a digital payment system ***nsucoin*** to buy from each other different sorts of flowers. Alice sells only chamomiles, Bob — only tulips, Caroline — only gerberas and Daniel — only roses. At the beginning each person has 5 flowers. The cost of each flower is 2 coins.

Transactions are used to make purchases by transferring coins in the system *nsucoin*. Each transaction involves two different users (the seller A and the buyer B) and distributes a certain amount of coins S between A and B , say $S = S_A + S_B$. The value S is equal to the sum of all the coins received by the buyer in the indicated k transactions, $1 \leq k \leq 2$. We will say that the current transaction is *based* on these k transactions. The value S_A is the amount of coins that the buyer pays the seller for his product, $S_A > 0$; the value S_B is the rest of available amount of coins S that returns to buyer (in further transactions B can spend these coins). At the same time, coins received by users in each transaction can not be distributed more than once in other transactions.

In order for transactions to be valid they must be verified. To do this **block chain** is used. Each block verifies from 1 to 4 transactions. Each transaction to be verified can be based on already verified transactions and transactions based on verified transactions.

There are 4 *special* transactions. Each of them brings 10 coins to one user. These transactions do not based on other transactions. The first block verifies all special transactions.

Define what bouquet Alice can make from the flowers she has if the last block in chain is the following string (hash of this block in 00004558):

`height:2;prevHash:0000593b;ctxHash:8fef76cb;nonce:17052`



Turn to the next page.

Technical description of *nsucoin*.

- **Transactions.** Transaction is given by the string `transaction` of the following format:

```
transaction = "txHash:{hashValue};{transactionInfo}"
hashValue = Hash({transactionInfo})
transactionInfo = "inputTx:{Tx};{sellerInfo};{buyerInfo}"
Tx = "{Tx1}" or "{Tx1,Tx2}"
sellerInfo = "value1:{V1};pubKey1:{PK1};sign1:{S1}"
buyerInfo = "value2:{V2};pubKey2:{PK2};sign2:{S2}"
```

Here `Tx1`, `Tx2` are values of the field `txHash` of transactions which the current transaction based on. V_i is a non-negative integer that is equal to the amount of coins received by the user with public key PK_i , $0 \leq V_i \leq 10$, $V_1 \neq 0$. Digital signature

$$S_i = \text{DecToHexStr}(\text{Signature}(\text{Key2}, \text{StrToByteDec}(\text{Hash}(\text{Tx1} + \text{Tx2} + \text{PK}_i))))$$

where `+` is concatenation operation of strings. `Key2` is private key of buyer.

In the special transactions fields `inputTx`, `sign1` are empty and there is no `buyerInfo`. For example, one of the special transactions is the following:

```
txHash:1a497b59;inputTx:;value1:10;pubKey1:11;sign1:
```

- **Block chain.** Each block is given by the string `block` of the following format:

```
block = "height:{Height};prevHash:{PrHash};ctxHash:{CTxHash};nonce:{Nonce}"
```

Here `Height` is the block number in a chain, the first block has number 0. `PrHash` is hash of block with number `Height - 1`. `CTxHash` is hash of concatenation of all the `TxHash` of transactions verified by this block. `Nonce` is the minimal number from 0 to 40000 such that block has hash of the form `0000####`.

Let `PrHash = 00000000` for the first block.

- **Hash function.** `Hash` is calculated as reduced MD5: the result of hashing is the first 4 bytes of standard MD5 represented as a string. For example, `Hash("teststring") = "d67c5cbf"`, `Hash("1a497b5917") = "e0b9e4a8"`.

- **Digital signature.** `Signature(key, message)` is RSA digital signature with n of order 64 bits, $n = 9101050456842973679$. Public exponents `PK` of users are the following:

User	Alice	Bob	Caroline	Daniel
PK	11	17	199	5

For example, `Signature(2482104668331363539, 7291435795363422520) = 7538508415239841520`.

- **Additional functions.** `StrToByteDec` decodes a string to bytes that are considered as a number. Given a number `DecToHexStr` returns a string that is equal to the hexadecimal representation of this number. For example, `StrToByteDec("e0b9e4a8") = 7291435795363422520` and `DecToHexStr(7538508415239841520) = "689e297682a9e6f0"`.

Strings are given in UTF-8.

Turn to the next page.

Examples of a transaction and a block.

• Suppose that Alice are buying from Bob 2 tulips. So, she must pay him 4 coins. The transaction of this operation, provided that Alice gets 10 coin in the transaction with hash 1a497b59, is

```
txHash:98e93fd5;inputTx:1a497b59;value1:4;pubKey1:17;sign1:689e297682a9e6f0;  
value2:6;pubKey2:11;sign2:fec9245898b829c
```

• The block on height 2 verifies transactions with hash values (values of txHash) 98e93fd5, c16d8b22, b782c145 and e1e2c554, provided that hash of the block on height 1 is 00003cc3, is the following:

```
height:2;prevHash:00003cc3;ctxHash:9f8333d4;nonce:25181
```

Hash of this block is 0000642a.