13^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

15 October 2011

Problem E - Yellow Balloon
Basis vectors

Problem Description

Your colleague has devised an (admittedly hare-brained) encryption scheme, and has asked you to implement an independent decryption routine. The scheme is based on the observation that some functions can be decomposed into a linear combination of given set of basis functions. If these basis functions are chosen carefully, this decomposition is unique; the actual basis functions also serve as the secret key in the encryption scheme.

For simplicity, your colleague has chosen the basis functions to be the polynomials of degree 0 through 10, i.e.,

$$\mathcal{B} = \{1, x, x^2, x^3, \ldots, x^9, x^{10}\}.$$

Next, the encryption algorithm takes a number p in the range 1..10 to encode. The first step is to form a linear combination of the basis functions such that only p of the basis functions have non-zero, integer coefficients, i.e., we want to end up with something like $f(x) = 2 + 4x^2 - 3x^4$. To construct $f(x)$, the encryption algorithm randomly generates a vector of coefficients $\mathcal{C}$; the details of this part of the algorithm are not important, but $\mathcal{C}$ has the following properties:

1. $\mathcal{C}$ contains 11 integer values, of which only p are non-zero;
2. The positions of the p non-zero coefficients are random;
3. The non-zero coefficients are in the range -10 through -1 or 1 through 10.

For example, the algorithm might produce $\mathcal{C} = \{2, 0, 4, 0, -3, 0, 0, 0, 0, 0, 0\}$. Let $f(x)$ then be the inner product of $\mathcal{B}$ and $\mathcal{C}$, thus

$$\begin{aligned} f(x) &= 2 \cdot 1 + 0 \cdot x + 4 \cdot x^2 + 0 \cdot x^3 - 3 \cdot x^4 + 0 \cdot x^5 + 0 \cdot x^6 + 0 \cdot x^7 + 0 \cdot x^8 + 0 \cdot x^9 + 0 \cdot x^{10} \\ &= 2 + 4x^2 - 3x^4 \end{aligned}$$

Observe that, by construction, $f(x)$ thus has a unique decomposition in terms of the set of basis functions $\mathcal{B}$. The function $f(x)$ is then used to generate a sequence of 11 pairs of coordinates of the form $\mathcal{S} = \{(x_i, f(x_i))\}$ for $i \in 0..10$. The choice of x_i can be arbitrary, but to reduce the chances of numerical overflow, they are chosen in the interval $[-2.5, 2.5]$ such that $x_k = -2.5 + 0.5k$.

The beauty (in the eye of the beholder, etc.) of this technique is that the sequence $\mathcal{S}$ can be used as the encrypted representation of the number p. Provided that the receiver knows which set of functions were used during encryption, he can decode the encrypted message $\mathcal{S}$ to recover the number p. Your task will be to implement such a decryption algorithm.

Input

Your input consists of an arbitrary number of records, each record conforming to the following format:

n
x_0 y_0
x_1 y_1
$\dots$ $\dots$
x_{n-1} y_{n-1}

The value n denotes the number of pairs of x_i and y_i values that will follow. Note that n will always be exactly 11 for valid records; it is included in the input merely to cleanly separate different records. The x_i and y_i pairs thus form a set $\mathcal{S} = \{(x_i, f(x_i))\},\ i \in 0..10$, as shown above.

Using only the knowledge that the 11 basis functions are the set $\mathcal{B} = \{1, x, x^2, x^3, \dots, x^9, x^{10}\}$, and the 11 (x_i, y_i) pairs forming $\mathcal{S}$, you must determine p, which is effectively the number of nonzero coefficients in $\mathcal{C}$, where $f(x) = \mathcal{C} \cdot \mathcal{B}$.

The end of input is indicated by a line containing only the value -1, equivalent to $n == -1$.

Output

For each input record, print out the line

`encoded number was` p

where p denotes the number of nonzero coefficients in the basis function decomposition.

Sample Input

```
11
-2.50 5.803412304688e+04
-2.00 6.408000000000e+03
-1.50 4.090449218750e+02
-1.00 2.000000000000e+01
-0.50 8.138671875000e+00
0.00 8.000000000000e+00
0.50 8.029296875000e+00
1.00 1.800000000000e+01
1.50 4.128417968750e+02
2.00 6.536000000000e+03
2.50 5.886420117188e+04
11
-2.50 -2.304687500000e+02
-2.00 0.000000000000e+00
-1.50 2.109375000000e+01
-1.00 7.000000000000e+00
-0.50 6.562500000000e-01
0.00 0.000000000000e+00
0.50 -7.812500000000e-01
1.00 -1.500000000000e+01
1.50 -1.122187500000e+02
2.00 -5.120000000000e+02
2.50 -1.722656250000e+03
-1
```

Sample Output

```
encoded number was 5
encoded number was 3
```

Time Limit

60 seconds