

Programming Problems
16th Annual Computer Science Programming Contest
Department of Mathematics and Computer Science
Western Carolina University
April 5, 2005
Test Cases

Problem 1: Encryption

A company wants to transmit data over the telephone line, but they are concerned that their lines are tapped. All of their data is transmitted as four-digit integers. They have asked you to write a program that encrypts their data so that it may be transmitted more securely. Your program should read a four-digit integer and encrypt it as follows: Replace each digit by $((\text{digit} + 7) \bmod 10)$. Then, swap the first digit with the third, swap the second digit with the fourth, and print the encrypted integer.

An example session (input is italicized):

Enter a Four Digit Number: *6254*

The Encrypted Number is: *2139*

Test cases:

Enter **6254**

Response should be **2139**

Enter **2357**

Response should be **2490**

Enter **0000**

Response should be **7777**

Enter **9644**

Response should be **1163**

Enter **-5**

Program should generate an error message and give user another chance to enter

Enter **9999**

Response should be **6666**

Problem 2: Degree of Inversion

Compute the extent to which an array of integers is out of sorted order with sorted order meaning that the smallest index has the smallest value, the second index has the second smallest value, and so on until the largest index has the largest value. The metric for computing the degree of inversion is defined as follows. For each array index i count how many elements at larger indexes have values strictly smaller than the value of the element at index i . The degree of inversion is the sum of this count over all indexes. Notice that if the array is sorted, then the degree of inversion is 0. Your program output must look like the example output shown below. User input is shown in italics and program output is shown in boldface.

Example Program Run 1

Please enter the array values: *4 3 2 1*

The degree of inversion is: 6

Example Program Run 2

Please enter the array values: *1 2 3 4*

The degree of inversion is: 0

Test cases:

Enter **4 3 2 1**

Response:

The degree of inversion is: 6

Enter **1 2 3 4**

Response:

The degree of inversion is: 0

Enter **8 4 5 7**

Response:

The degree of inversion is: 3

Enter **4 5 7 3**

Response:

The degree of inversion is: 3

Enter **100 102 93 107**

Response:

The degree of inversion is: 2

Enter **1000 0 2000 2001**

Response:

The degree of inversion is: 1

Problem 3: Pseudo-Random Numbers

A common pseudo-random number generation technique is called the *linear congruential method*. If the last pseudo-random number generated was L , then the next number is generated by evaluating $(Z \times L + I) \bmod M$, where Z is a constant multiplier, I is a constant increment, and M is a constant modulus. For example, suppose Z is 7, I is 5, and M is 12. If the first random number (usually called the *seed*) is 4, then we can determine the next few pseudo-random numbers are follows:

Last Random Number, L | $(Z \times L + I)$ | Next Random Number, $(Z \times L + I)$
mod M

----- -----		

4	33	9
9	68	8
8	61	1
1	12	0
0	5	5
5	40	4

In this problem you will be given sets of values for Z , I , M , and the seed, L . Each of these will have no more than four digits. For each such set of values you are to determine the length of the cycle of pseudo-random numbers that will be generated. But be careful – the cycle might not begin with the seed!

Input

Each input line will contain four integer values, in order, for Z , I , M , and L . The last line will contain four zeroes, and marks the end of the input data. L will be less than M .

Output

For each input line, display the case number (they are sequentially numbered, starting with 1) and the length of the sequence of pseudo-random numbers before the sequence is repeated.

An example session

Sample Input

```
7 5 12 4
5173 3849 3279 1511
9111 5309 6000 1234
1079 2136 9999 1237
0 0 0 0
```

Sample Output

```
Case 1: 6
Case 2: 546
Case 3: 500
Case 4: 220
```

Test cases:

Enter **7 5 12 4**

Response: **6**

Enter **5173 3849 3279 1511**

Response: **546**

Enter **1079 2136 9999 1237**

Response:**220**

Enter **7 5 13 4**

Response: **12**

Enter **0 0 0 0**

Response: **Exit**

Problem 4: Zipper

Given three strings, you are to determine whether the third string can be formed by combining the characters in the first two strings. The first two strings can be mixed arbitrarily, but each must stay in its original order.

Test cases:

Enter **cat tree tcræte**

Response: **yes**

Enter **cat tree catrtee**

Response: **yes**

Enter **cat tree cttaree**

Response: **no**

Enter **bob ray brayob**

Response: **yes**

Enter **dead wood dwooeadd**

Response: **yes**

Enter **dead wood doeadwod**

Response: **no**

Enter **ray bob brayob**

Response: **yes**