2018 National Collegiate Programming Contest
Preliminary Round

September 29, 2018

- Problems: There are 7 tasks (12 pages in all) in this packet.
- Program Input: Input to the program are through standard input. Program input may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Program Output: All output should be directed to the standard output (screen output).
- Time Limit: Judges will run each submitted program with certain time limit (given in the table below).

Task Information

	Task Name	Time Limit
Problem A	Cover a Hole	1 sec.
Problem B	Longest Common Substrings	1 sec.
Problem C	Partition an Integer	1 sec.
Problem D	Simple Congruence Equation	1 sec.
Problem E	NCPE	1 sec.
Problem F	Counting the Machines	2 sec.
Problem G	Image Restoration	1 sec.

Problem A
Covering a Hole

Tom works in a company that produces covers for all kinds of holes, such as holes on streets and wells. He encounters a problem as follows: given a hole H which is a polygon with interior angles of only 90 or 270 degrees, determine the smallest rectangular cover that can completely cover H. In this problem, H is given in a coordinate system such that each of its edges is either vertical or horizontal. When covering a hole, each edge of the cover should also be either vertical or horizontal in the same coordinate system.

Consider the example in Figure 1. It is easy to see that the smallest rectangular cover that can completely cover H is a rectangle of size 4×8.

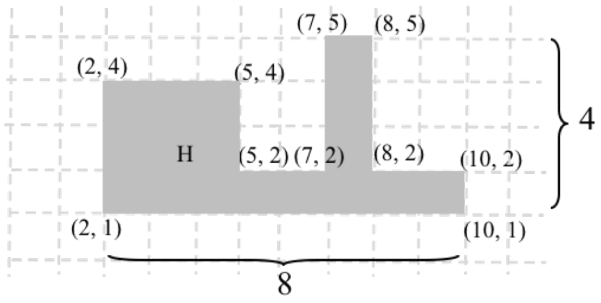

Figure 1: a rectangular hole H.

In this problem, you are asked to find the area of the smallest rectangular cover that can completely cover H. For example, in Figure 1, the output is 32.

Technical Specification

1. The number of the vertices of H, denoted by n, is a positive integer between 4 and 100.

2. The x-coordinates and y-coordinates of vertices are integers between 0 and 1000.

Input Format

The first line is an integer t, $1 \leq t \leq 10$, indicating the number of test cases. Each test case starts with one line containing the number n, $4 \leq n \leq 100$, of vertices of the hole H. Then, n lines follow, each of which contains two integers x and y, $0 \leq x, y \leq 1000$, which are the coordinates of the vertices of the hole's polygon in the order they would be visited in a trip around the polygon.

Output Format

For each test case, output the area of the smallest rectangular cover that can completely cover H in one line.

Sample Input

```
2
10
10 1
10 2
8 2
8 5
7 5
7 2
5 2
5 4
2 4
2 1
4
2 1
5 1
5 4
2 4
```

Sample Output for the Sample Input

```
32
9
```

Problem B
Longest Common Substrings

Given two nonempty 0-1 strings A and B, output the length of a longest common substring of A and B.

Technical Specification

1. There are at most 10 test cases.
2. Each of A and B has length no more than 100.

Input Format

The first line is an integer t, $1 \leq t \leq 10$, indicating the number of test cases. Each test case is specified simply by A and B, with a single whitespace character in-between on a single line.

Output Format

For each test case, output the length of a longest common substring of A and B.

Sample Input

```
3
001 10010
11111 0111101
000 111
```

Sample Output for the Sample Input

```
3
4
0
```

Problem C
Partition an Integer

Let n be an integer greater than 2. Write an efficient program to find 3 positive integers a, b, and c, such that $n = a + b + c$ and that their least common multiple, lcm(a, b, c), is as small as possible.

For example, $17 = 2 + 5 + 10$ and $\text{lcm}(2, 5, 10) = 10$. However, $17 = (1 + 8 + 8)$ and $\text{lcm}(1, 8, 8) = 8$ also is possible. Thus, in this problem, the division of 17 into $1, 8, 8$ is better than $2, 5, 10$, because $1, 8, 8$ has smaller least common multiple.

Input File Format

There are more than one test cases in the input file. Each test case contains an integer n in a line. The last test case is followed by a line containing 0. The value of n is greater than 2 and less than 2^{31}.

Output Format

For each test case, print out the values of a, b, and c. It is required that $a + b + c = n$, $0 < a \leq b \leq c$, and lcm(a, b, c) is minimized.

If the solution is not unique, print the solution with smallest a. If there are many solutions with the smallest a, print the one with smallest b.

Sample Input

```
12
17
25
0
```

Output for the Sample Input

```
4 4 4
1 8 8
5 10 10
```

Problem D
Simple Congruence Equation

Solving the linear congruence integer equation of $ax \equiv r \ mod \ P$ with $gcd(a, P) = 1$ (that is, a and P are coprime positive integers) is a fundamental problem in the study of Information Security.

Given positive integers a, r, and P such that $gcd(a, P) = 1$, write a program to find the positive integer $x \in (0, P)$ to satisfy $ax \equiv r \ mod \ P$.

Restriction: $0 < a, r < P < 2^{31}$, and $gcd(a, P) = 1$.

Input Format

The first line of the input contains one integer, $N \leq 8$, indicating the number of test cases to come. Each test case consists of three integers a, r, P in a line with a, r, P being separated by blank spaces.

Output Format

For each test case, output x for $ax \equiv r \ mod \ P$ on a single line.

Sample Input

```
3
4 3 5
3 2 256
65536 1 2147483647
```

Sample Output for the Sample Input

```
2
86
32768
```

Problem E
NCPE

NCPE is a recursive acronym for "NCPE is Collegiate Programming Examination". Here are the recursive rules:

1. N→NCPEis
2. C→Collegiate
3. P→Programming
4. E→Examination

Applying all the rules simultaneously on an initial string, we will get a new string. If we apply the rules again and again, the resulted string grows longer and longer. For exmaple, if we start with "NCPE", we get:

- NCPE →
- NCPEisCollegiateProgrammingExamination →
- NCPEisCollegiateProgrammingExaminationisCollegiateollegiateProgramming ···
- ···

However, we are not interested in the exact string. We just want ot know that after the rules are simultaneously applied recursively to an initial string S_0 a total of p^q times, whether a particular test character W would appear 'Odd' or 'Even' times in the resulting string. For example, after appling the above rules to the initial string "NCPE" just once, the character 'm' would appear 'Odd' (3) times.

Technical Specification

1. Only characters in the set $\{C, E, N, P, a, e, g, i, l, m, n, o, r, s, t, x\}$ will be used in this task.
2. The left side of each rule has exactly one character.
3. Each character will appear on the left side of at most one rule.
4. If a character c does not appear on the left side of any rule, a default rule $c \to c$ is assumed.
5. The initial string contains at most 32 characters.
6. The length of any rules is at most 20.
7. $1 \leq p \leq 10^{50}$.
8. $1 \leq q \leq 10^{50}$.

Input Format

The first line is an integer (≤ 20) indicating the number of test cases to follow. The first line of each test case is an integer M indicating the number of rules. Each of the next M lines contains one rule written in the format "x→S" where x is a character and S is a sequence of characters. Following the rules, the initial string S_0, the test character W, and numbers p, q are given on a single line.

Output Format

For each test case, if the test character W appears odd times in the final string, output "Odd", otherwise output "Even".

Sample Input

```
3
4
N->NCPEis
C->Collegiate
P->Programming
E->Examination
NCPE m 1 1
4
N->NCPEis
C->Collegiate
P->Programming
E->Examination
NCPE m 2 3
4
C->xeCtlonPlmmC
E->eetat
N->tEalxatNsisrmimE
P->tEeEEngoxmonoi
CN P 1 1
```

Sample Output for the Sample Input

```
Odd
Even
Odd
```

Problem F
Counting the Machines

There are a number of machines working together in a factory. Machines assemble parts and transfer parts to other machines via unidirectional links between them. A factory of scale $n(3 \leq n \leq 8)$ contains $n!$ machines and each of the machines is labeled with a unique permutation of n distinct digits from 1 to n. Every machine has $n-1$ outgoing links connecting $n-1$ outgoing neighbors. Let machine $S = s_1 s_2 s_3 \ldots s_n$. The outgoing neighbors of S are $s_2\mathbf{s_1}s_3 \ldots s_n$, $s_2 s_3 \mathbf{s_1} \ldots s_n$, ... and $s_2 s_3 \ldots s_n \mathbf{s_1}$, which are obtained by inserting the first digit of S to the k^{th} position, where $2 \leq k \leq n$. Figure 1 shows a factory of scale 3. The outgoing neighbors of machine 123 are 213 and 231. Moreover, in a factory of scale 8, the outgoing neighbors of 12345678 are 2**1**345678, 23**1**45678, 234**1**5678, 2345**1**678, 23456**1**78, 234567**1**8 and 2345678**1**.

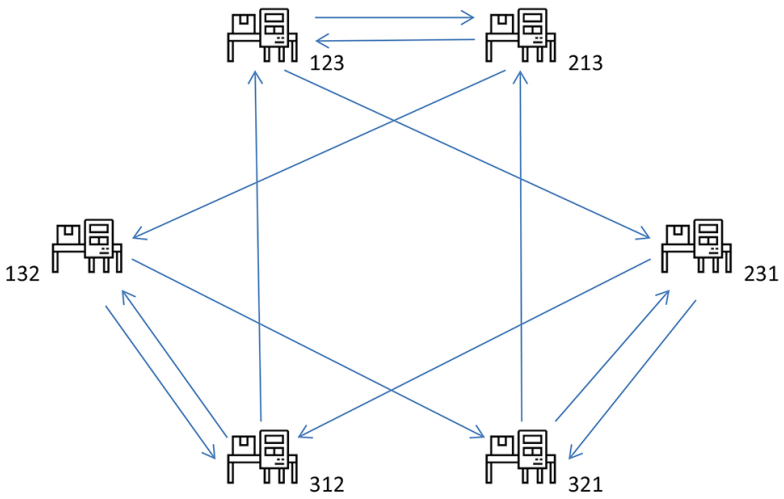

Figure 1: A factory of scale 3

There are different paths from a given source machine to a given destination machine. Only one of the paths has the minimum number of links, which is the shortest path. In addition, the number of links in a shortest path is the shortest length. For example, from 123 to 312, the shortest path is $123 \rightarrow 231 \rightarrow 312$ and the shortest length is 2.

Let X and Y be two machines. We denote $path(X, Y)$ as the set of machines included in the shortest path from X to Y. We also denote $length(X, Y)$ as the shortest length from X to Y. For example, $path(123, 312) = \{123, 231, 312\}$ and $length(123, 312) = 2$.

The problem is as follows: Given the scale of a factory (n), the permutation of a source machine (X), the permutation of an intermediate machine (M) and a shortest length (d), please count the machine (Y) in a factory of scale n that satisfies the following requirements:
1. $length(X, Y) = d$, and
2. $M \in path(X, Y)$.

Input Format

The first line of the input is an integer indicating the number of test cases to follow. There are at most 5 test cases. A test case has 4 items in one line. Two items are separated by a space. The first item is an integer $n(3 \leq n \leq 8)$ indicating the scale of a factory. The second item is the permutation of a source machine. The third item is the permutation of an intermediate machine. The last item is an integer $d(2 \leq d \leq n-1)$ indicating the shortest length.

Output Format

For each test case, output an integer indicating the number of machines in a given factory satisfying the requirements.

Sample Input

```
2
3 123 231 2
4 4231 2314 3
```

Sample Output for the Sample Input

```
2
8
```

Problem G
Image Restoration

A fax machine scans a page and electrically breaks up a document into picture elements (pixels). Each pixel has two possible values: black (stored as 0) or white (stored as 1). A received fax is often noisy: some black pixels in the original image were flipped to white pixels, and some white pixels were flipped to black. For example, Fig. 1 (left) shows a scanned image printing a capital A. We would like to restore it to Fig. 1 (right).

The image restoration task is, in fact, a process of solving an optimization problem. In the restored image, each pixel should also be either black (0) or white (1). The restoration process considers two aspects:

1. We want to keep original content—if a pixel is black in the original image, it is more likely to be black in the restored image; and a white pixel in the original image is more likely to be white in the restored image.

2. We want to clean up the original image—black pixels tend to group together in the restored image, so do white pixels.

Therefore, two types of costs for restoring an image are defined as follows:

1. If a pixel changes its value in the restored image, the type-1 cost of fixing the pixel is 2. For example, the type-1 cost of restoring Fig. 2 (left) to Fig. 2 (right) is 2, because one pixel (marked with a star symbol) changes its value.

2. The type-2 cost counts the number of discontinuities in the restored image, regardless of the original image. The type-2 cost of Fig. 2 (right) is 3, because there are 3 pairs of neighboring pixels (under a standard 4-neighborhood system) have different values, marked with dash lines.

The total cost of a restored image is the sum of type-1 and type-2 costs, and a good restoration should have a low cost. The restoration cost of Fig. 2 (right) is 5, which is not the best restoration because the minimum cost of this example is 4, considering that we do nothing to the original image.

Please write a program to compute the minimum cost of restoring an input image.

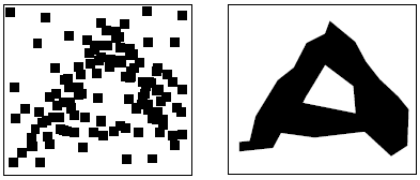

Figure 1: (left) original image (right) restored image.

Input Format

The input contains several test cases. For each test case, the first line contains two integers H and W ($1 \leq H, W \leq 10$) indicating the resolution of the input image. The next H lines each contains W pixel values: 0 (black) or 1 (white). The zero value of H and W indicate the end of test cases and should not be processed.

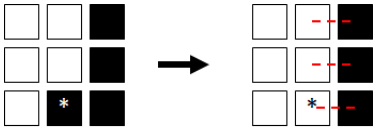

Figure 2: Example illustrating two types of restoration costs.

Output Format

For each test case, please output the minimum restoration cost.

Sample Input

```
3 3
1 1 0
1 1 0
1 0 0
3 3
1 0 1
0 1 0
1 0 1
0 0
```

Sample Output for the Sample Input

```
4
8
```