Problem A
Verify Collatz $3x + 1$ Conjecture
Time limit: 1 seconds

Given a positive integer x_0. Consider the sequence x_0, $x_1, x_2, \ldots$, where

$$x_i = \begin{cases} x_{i-1}/2, & \text{if } x_{i-1} \text{ is even} \\ 3x_{i-1} + 1, & \text{if } x_{i-1} \text{ is odd} \end{cases} \qquad \text{for } i > 0.$$

This sequence is, in general, infinite. In 1937, German mathematician Lothar Collatz conjectured that this sequence will eventually reach the number 1, regardless of which positive integer is chosen initially. For example, if $x_0 = 13$, then the sequence is

$$13, 40, 20, 10, 5, 16, 8, 4, 2, 1.$$

If $n = 1$, the sequence is

$$1, 4, 2, 1.$$

Given a positive integer x_0, write a program to compute the sequence $x_1, x_2, \ldots$, until $x_m = 1$ first appears. Print out the index m where $x_m = 1$, and the maximum value of the sequence.

Input Format

Each test case contains only 1 positive integers x_0 in one line. The value of x_0 is greater then 0 and less than 2^{32}.

Note that the test data file may contain many test cases. The last test case is followed by a line containing a single 0.

Output Format

The outputs for each test case should be the numbers x_0, followed by 2 integers m and l, where m is the smallest positive index with $x_m = 1$, and l is the maximum value of the sequence. For example, for $x_0 = 13$, print out
13 9 40
since $x_9 = 1$, and the maximum value of the sequence is 40.

If the sequence never reach 1 within 2^{24} steps, print out two 0's after x_0.

Sample Input

```
13
1
0
```

Sample Output for the Sample Input

```
13 9 40
1 3 4
```

Problem B
Largest Three Companions
Time limit: 1 seconds

Karen is studying particle collision reaction, where each particle has an integral score, positive or negative. The scores range from -1000 to 1000. When particles collide, the scores of involved particles are multiplied. Karen wants to find out which three of the particles can yield the largest product. Write a program to find the largest product.

Input Format

The first line of the input gives the number of test cases, T ($T \leq 5$). For each test case, the first line consists of one positive integer N ($3 \leq N < 1000$), indicating the number of particles. Then follows a line with N integers, separated by space(s), ranging from -1000 to 1000.

Output Format

For each test case, output the answer in a separate line.

Sample Input

```
2
3
1 3 2
4
1 -3 1 5
```

Sample Output for the Sample Input

```
6
5
```

Problem C
Data Recovery
Time limit: 2 seconds

In the IoT era, mass data transimision and exchanges occur every second. To prevent secret information or documents from an accidental loss or an intentional release, the concept of information sharing and recovering has raised an interest of a lot of researchers. This problem asks you to write a program to solve a data recovery problem based on a collection of a sufficient number of participants according to modular operations of Number Theory.

Let P be a large prime number, for example, $P = 65537$, suppose that the secret message has been encryted and converted into M pairs of integers $(x_i, y_i),\ i = 1, 2, \cdots, M$, to allocate to M participants according to the following strategy

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + \cdot + a_{n-1} x_i^{n-1} \ \ (mod\ P)$$

where the secret message is represented as the integers $a_0, a_1, a_2, \cdots, a_{n-1}$, and each participant holds partial information $(x_i, y_i),\ i = 1, 2, \cdots, M$ about the whole secret message where $n \leq M < P$ and all of the computations are under modular P operations.

According to the strategy of secret sharing and recovering as introduced above, collecting any n distinct pairs of integers (x_j, y_j) from M participants, one can recover the original message $a_0, a_1, \cdots, a_{n-1}$. This problem asks you to write a program to reconstruct the secret message $a_0, a_1, \cdots, a_{n-1}$ based on n given pairs of integers $\{(x_j, y_j)|\ j = 1, 2, \cdots, n\}$ under the modular P computations.

Restriction: $P = 65537,\ M = n = 3$ are fixed and $a_0, a_1, a_2 \in [0, P)$, $0 < x_1 < x_2 < x_3 < P$ in this problem.

Input Format

The first line of the input file contains one integer $K \leq 5$ indicating the number of test cases to come. *Each of the test cases* consists of 3 lines of the information

$$x_1 \quad y_1$$

$$x_2 \quad y_2$$

$$x_3 \quad y_3$$

where x_i and y_i are separated by a Tab or spaces and $0 \leq x_i, y_i < P,\ i = 1, 2, 3$.

Output Format

There are K lines of the output file, each line contains a triple of integers

$$a_0 \quad a_1 \quad a_2$$

for each test case, where $0 \leq a_0, a_1, a_2 < P$ which are separated by a Tab or spaces.

Sample Input

```
2
1 4
2 9
3 16
4 147
8 515
16 1923
```

Sample Output for the Sample Input

```
1 2 1
3 8 7
```

Problem D
Routing mode checking
Time limit: 5 seconds

The mesh-based Network-on-Chip (NoC) has been viewed as a practical solution for the interconnection in the multi-core systems. The data communication among each core by following the predefined data delivering routing algorithms. The routing algorithm can be classified into XY routing algorithm and adaptive routing algorithm. The XY routing algorithm always transmits the data to the target core by following one x-axis direction delivery first and then one y-axis direction delivery, as shown in Figure 1(a). On the other hand, the adaptive routing algorithm provides more routing path diversity than the XY routing algorithm if each core in the minimal routing region is active, as shown in Figure 1(b). Therefore, each core in the minimal routing region is defined as the adaptive routable core. The minimal routing region is a rectangle region bounded by the source core and the destination core, as shown in Figure 1(b). However, some cores will become inactive core because of system reliability issue. In this case, some cores may become non-reachable cores for some source cores. Figure 1(c) illustrates an example to indicate the adopted routing mode (*i.e.*,XY routing algorithm or adaptive routing algorithm) for each core as identifying the source core.

Given the location of a source core and the 8-by-8 NoC topology status. Therefore, there are 64 cores in this NoC multi-core system. You need to find a way to decide the proper routing mode to deliver the data to every other reachable core. If the core is non-reachable core, you need to indicate it.

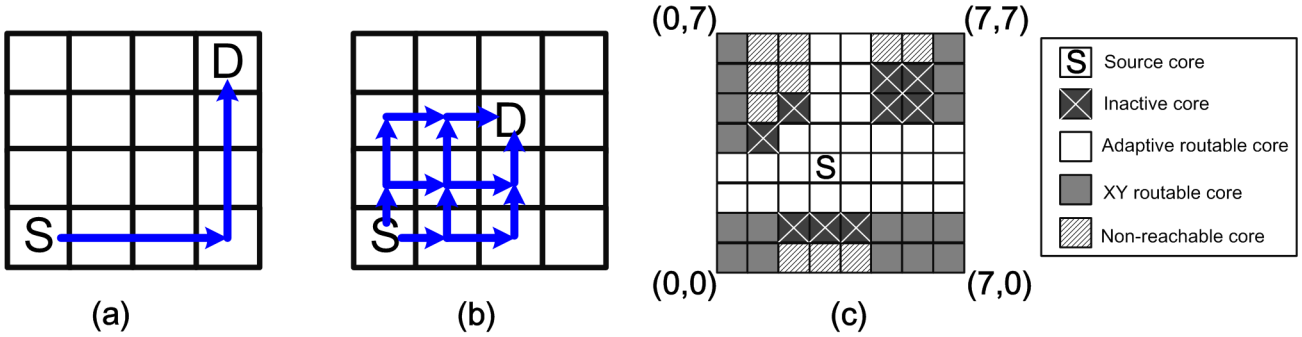

Figure 1: a)The XY routing algorithm ,(b)the adaptive routing algorithm, and (c)the example routing mode checking.

Input Format

An instance of the problem consists of 64 numbers. The first number represents the top left (0,7) core, the second number represents the (1,7) core, ,and the last number represents the

bottom right (7,0) core. If the number is 0, it is the inactive core. If the number is 1, it is the active core. If the number is 2, it is the source core. Note that the test data file may contain more than one instances. The last instance is followed by a line containing a single 0.

Output Format

The output must contains 64 numbers for each instance. The first number represents the top left (0,7) core, the second number represents the (1,7)core, ,and the last number represents the bottom right (7,0) core. If the number is 0, it is the adaptive routable core. If the number is 1, it is the XY-routable core. If the number is 2, it is the inactive core. If the number is 3, it is the non-reachable core. If the number is 4, it is the source core. If there are multiple states for a core, the priority is $4 > 2 > 0 > 1 > 3$.

Sample Input

```
0000000000000110001001100100000000020000000000000011100000000000
0
```

Sample Output for the Sample Input

```
1330033113300221132002211200000000040000000000001122211111333111
```

Problem E
Terrestrial Stations
Time limit: 3 seconds

Problem Description

In a state, there are many towns. To adapt to the digital age, the governor wishes to provide digital television service to each town by setting up terrestrial stations (also known as digital broadcast stations) in some of the towns. Assuming that the service range is the same for every terrestrial station, that is, any person within distance R from a terrestrial station will be able to receive digital television signals. For ease of description, the location of each town is represented by a 2-dimensional coordinate. For instance: there are five towns in the state, represented by P_1, P_2, P_3, P_4, and P_5. Their coordinates are (1, 3), (5, 3), (5, 6), (7, 4), (8, 6), respectively. When $R = 3$, by setting up terrestrial stations at P_1 and P_4, we can provide television signals to every town (see Figure 1). When $R = 5$, setting up one terrestrial station at P_3 alone is enough (see Figure 2).

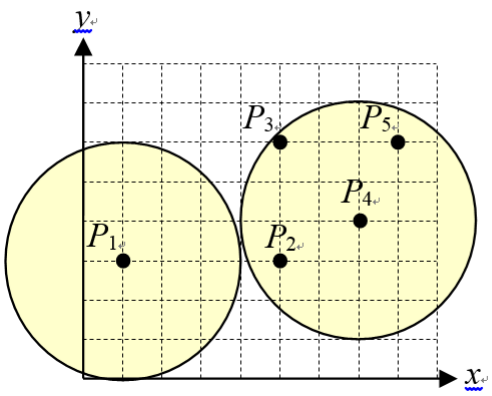

Figure 1: $R = 3$.

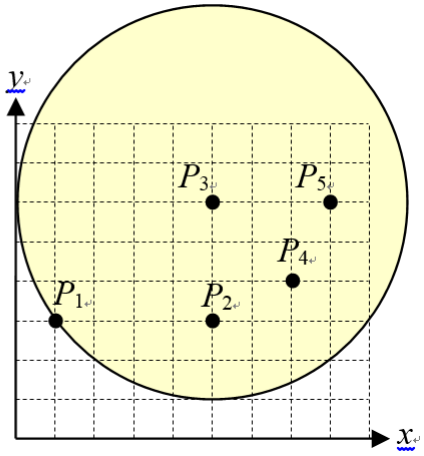

Figure 2: $R = 5$.

In this problem, you are given the coordinates of N towns $P_1, P_2, \ldots, P_N$, and two positive integers R and k, where $1 \leq R \leq 10000$ and $1 \leq k \leq N \leq 22$. Given the coordinates of all towns in the state (every town has a different coordinate), please determine if it is possible to provide digital television signals to all towns by setting up at most k terrestrial stations at the towns, and output the selected towns.

Technical Specification

- The number of test cases is at most 10.
- The number N of towns is an integer between 1 and 22.
- The x- and y-coordinate (x_i, y_i) for each town are integers between 1 and 10000.
- The range R of the terrestrial stations is an integer between 1 and 10000.
- The positive integer k is between 1 and N.

Input Format

The first line of each test case gives the 3 integers N, R, and k ($1 \leq N \leq 22$, $1 \leq R \leq 10^4$, and $1 \leq k \leq N \leq 22$), where N represents the number of towns, R represents the range of each terrestrial station, and k represents the maximum number of terrestrial stations. The first line is followed by N lines which describe the integer coordinates of the N towns, where the ith line gives x_i and y_i of town P_i (where $1 \leq x_i, y_i \leq 10000$). The input is terminated by a line containing three zeros, which should not be processed.

Output Format

For each case, if it is possible to provide digital television service to all towns by setting up at most k terrestrial stations, output the minimum number of terrestrial stations required in the first line, and the towns to set the terrestrial stations in the next line (output the stations in increasing order, separate each base station with a space). If there are multiple possible ways to set up the terrestrial stations, output the one with the smallest lexicographic order. If it is not possible to provide service by setting up at most k terrestrial stations, output -1.

Sample Input

```
5 3 3
1 3
5 3
5 6
7 4
8 6
5 3 1
1 3
5 3
5 6
7 4
```

```
8 6
0 0 0
```

Sample Output

```
2
1 3
-1
```

Problem F
Battle Calculator
Time limit: 3 seconds

In most of the fighting games, players can use a variety of moves to knock down opponents. The player wins when the opponent's blood (HP) dropped to 0 or below. When these moves, according to a certain order (combo), cause a lot of damage and make it impossible for the opponent to fight back, the player could get a quick victory. In order to make the game diverse, the game designer limited the moves to be launched according to previous move. For example, if the opponents blood is 1000, you have 3 moves ($fistpunch, kick, poke$) to choose from. $Fistpunch$ can knock down opponents blood for 150 and it takes 200 millisecond to launch, the move that can be followed is $kick$. $Kick$ can knock down opponents blood for 500 and it takes 1000 millisecond to launch, the move that can be followed is $fistpunch$. $Poke$ can knock down opponents blood for 20 and it takes 20 millisecond to launch, the move that can be followed is $poke$. To knock down the opponent, if the move orders are $fistpunch, kick, fistpunch$, then $kick$, which cause 1300 points of damage in 2400 millisecond, it is not as good as $kick, fistpunch$, then $kick$ because the latter one cause 1150 points of damage in only 2200 milliseconds. Now you are offered the moves data of one character in a game, please calculate a set of the fastest solution to beat up your opponent. In the case of tie of time, sort the output sequence of moves as string (blank between each move is included) and output the one with smallest lexical order.

Input Format

The first line of the input is a positive integer n ($n \leq 20$) indicating the number of test data. The first line of each data has two integers s ($0 < s \leq 30$) and HP ($0 < HP < 11000$) represent total number of moves and the components blood (HP) left at present respectively. Then, it is followed by s lines, each line contains the following information separated by a space.

- The name of the move (no more than ten characters without any space)
- The damage the move can cause ($0 < damage < 1200$). Each damage is supposed to reduce the components HP.)
- The time in millisecond (< 1050) needed for each move.
- The names of the next possible moves after the previous move (at least one and at most five, separating by a space if there are more than one.)

Output Format

Each test data output two lines. The first line is the moves sequence (each move separated by a space). The second line contains two integers, the total time needed to launch the moves, then the damage caused. If there is no way to beat it up or more than ten moves are needed, please output "impossible" (no quotation marks) in the first line and two zero in the second line.

Sample Input

```
3
3 1000
fistpunch 150 200 kick
kick 500 1000 fistpunch
poke 20 20 poke
2 2000
poke 20 20 poke push
push 20 20 push poke
2 1000
AAA 250 500 AAA
BBB 500 1000 BBB
```

Sample Output for the Sample Input

```
kick fistpunch kick
2200 1150
impossible
0 0
AAA AAA AAA AAA
2000 1000
```

Problem G
Chaining
Time limit: 2 seconds

Lucy is playing a treasure hunting game with her friends. They have to decode a encoded message to receive hints. The encoded message is given in a sequence of numbers between *0* and *n*. To decode the message, Lucy and friends need to use each given number as pointer to point to the position in the seuqence as indicated by the given number. For example, in Figure 1(a), the given encoded message is *5 4 0 6 3 1*. The first number, which is *5*, act as pointer that points to the fifth number in the encoded message, which contains the number *3*. The fifth number, which is *3*, then act as a pointer that points to the third number in the encoded message. Figure 1(b) depicts the final message chain. Note that *0* denotes the end of the message, so it does not point to any other position in the encoded message. Given this chain, the decoded message is the list of position numbers of the chain from the head of the chain, which is position 2 (that contains number *4*), to the end of the chain, which is position 3 (that contains number *0*). So the final decoded message is *2 4 6 1 5 3*. In this instance, there is only one message chain, but there can be more than one message chain in an encoded message.

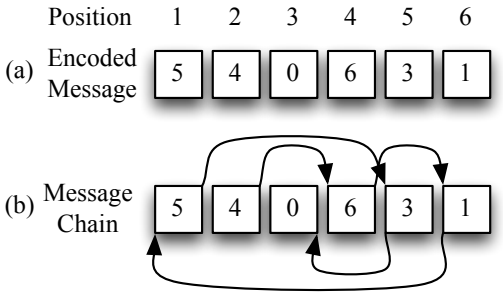

Figure 2: (a) Encoded message. (b) Message chain

Please write a program to help Lucy and friends to decode messages.

Technical Specification

1. The length of the encoded message is n, $1 \leq n \leq 10000$.
2. There are at least 1 and at most 15 message chains in an input instance.
3. Each position in the encoded message must belong to exactly one message chain.

Input Format

An instance of the problem consists of two lines of input. The first line contains a single integer n, denoting the length of the encoded message. The second line contains encoded message which has n integers between *0* and n. There is always a space between any two consecutive integers. The last instance is followed by a line containing a single *0*.

Output Format

For each instance, output several integers. The first integer is the number of message chains contained in the given encoded message. For each message chain, output the starting position and the length of the message chain. There should be one space between any two consecutive integers. If there is more than one message chain, output in the order of the starting positions of the message chains.

Sample Input

```
6
5 4 0 6 3 1
12
0 6 0 10 7 4  3 1 12 5 2 11
3
0 0 2
0
```

Sample Output for the Sample Input

```
1 2 6
2 8 2 9 10
2 1 1 3 2
```

Note that the first output
1 2 6
means 1 message chain, the chain starts at position 2 and has length 6.

The second output
2 8 2 9 10
means 2 message chains, the first one starts at position 8 and has length 2, and second one starts at position 9 and has length 10.