Problem A- Paperboy

A paperboy wants you to help him optimize his delivery route, which is on one side of a [very] long, linear street. He spends time moving between houses and time delivering papers. The time to move between a pair of houses equals the difference in the house numbers. The time to deliver a paper is 1.

The papers are dropped off in front of house number x. What's the smallest amount of time it will take him to deliver the papers?

Input Specification:

You will be presented with several test cases composed only of positive integers, one per line. Each line will begin with the number $n < 10000$, the number of houses, followed by $n + 1$ house numbers. The first n of these are where the paperboy delivers his papers, the last house number is where he starts his paper route. House numbers never exceed 100000.

The input ends on EOF.

Output Specification:

For each test case, output the minimum amount of time it takes for the paperboy to finish his route.

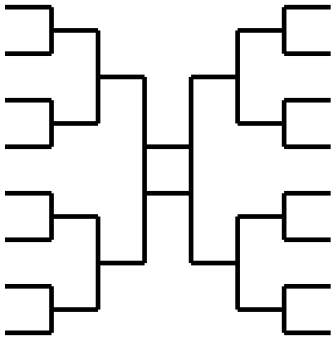
Sample Input:

```
3 10 20 30 10
4 10 20 40 80 30
```

Sample Output:

```
23
94
```

Problem B- Repechage



A *repechage*, a word from the French meaning "re-fishing", is used by tournament organizers to allow competitors who lost by a small margin to advance to the next round.

In martial arts athletics, the two finalists are determined by a single elimination tournament. The winner of the final wins the gold, the loser wins the silver.

The repechage is used to determine the bronze medal. To qualify for the repechage, a competitor must have lost to one of the finalists.

Input Specification:

The input are a sequence of test cases. Each test case begins with the integer N. When N is 0, that is the signal that the input has ended. Otherwise, $2 \leq N \leq 6$.

Next come $2^N - 1$ lines with pairs of names separated by whitespace, i.e., $name_1$ $name_2$. Names are composed of alphabetic characters only, and will never exceed 10 characters. The meaning of each line of input is that $name_1$ defeats $name_2$.

The data always describe a perfect single elimination tournament bracket, in which half of the competitors are eliminated in round 1, half of those remaining are eliminated in round 2, and so forth, until a single winner remains. There are always exactly 2^N distinct names.

Output Specification:

For each case, output the winner of the "`Gold: `", the "`Silver: `" and the qualifiers for the "`Repechage: `", in a space-separated, alphabetical ordered list.

Sample Input:

```
3
A B
C D
F E
H G
A C
H F
A H
0
```

Sample Output:

```
Gold: A
Silver: H
Repechage: B C F G
```

Problem C- Happy Separation

There are 3 happy people and 3 unhappy people standing at distinct points of a very large field. Happiness is contagious and misery loves company, so the ideal situation for everyone is to build a straight line wall that separates them.

Input Specification:

Input is presented two lines per test case, ending with EOF. The first line $x_1\ y_1\ x_2\ y_2\ x_3\ y_3$ lists the coordinates of the happy people, and the second line $x_4\ y_4\ x_5\ y_5\ x_6\ y_6$ the unhappy people. All coordinates are between ± 1000, and no three people are collinear.

Output Specification:

Output "`Yes`" if a wall can be built to separate them and "`No`" if no such wall exists.

Sample Input:

```
1 1 2 2 2 3
1 0 5 1 4 5
1 0 -2 1 -2 -1
-1 0 2 1 2 -1
```

Sample Output:

```
Yes
No
```

Problem D- Coin Stacks

You have several stacks of identical coins. You wonder how you can combine them all into one big stack, using a sequence of the following two operations:

- **Combine** - For two stacks of exactly equal height, combine them together to form a single stack.
- **Difference** - For two stacks of unequal height, take out the difference and treat it as a new stack. The rest of the coins are combined into one single stack. In other words, if there are two piles with a and b coins each, where $a > b > 0$, replace them with piles of size $2b$ and $a - b$.

For example, if you have piles with sizes 1, 3, 4, you could perform the difference operation on the first two piles (getting piles with sizes 2, 2, 4), then combine the two piles with size 2, and finally combine the remaining two piles to get a single pile with size 8.

Input Specification:

The first line contains the integer $T \leq 100$, followed by the data for T test cases. Each test case begins with a line containing integer N, $2 \leq N \leq 200$, which represents the number of stacks. The next line contains N positive integers, separated by spaces, which represent the sizes of the stacks, such that the total number of coins equals 2^K for some integer $K \leq 30$.

Output Specification:

For each test case, give a correct sequence of no more than 10000 steps that combine all the coins. For each step, put a pair of space-separated integers describing the sizes of the stacks on which you perform the operation. If the integers are equal, it is a combine operation; otherwise it is a difference operation. After the last operation of each test case, output "`0 0`".

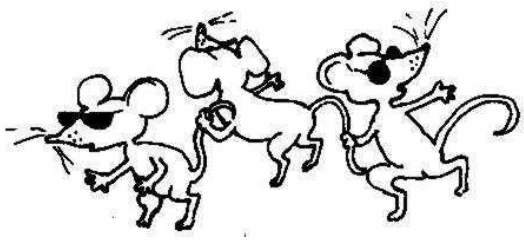
Sample Input:

```
1
3
1 3 4
```

Sample Output:

```
1 3
2 2
4 4
0 0
```

Problem E- Blind Maze



There is a grid consisting of 3x3 cells, of which K contain boulders. One cell contains the starting point and another different cell contains the exit.

You can move in any of the 4 compass directions: North, East, South, West. If you try to move into a cell containing a boulder, or move out of the grid, you fail and stay at the same place. You want to move from the starting point to the exit.

Sound easy?

The problem is, the grid is completely dark:

- You stand at the starting point, but do not know its exact location.
- You do not know where the exit is.
- You do not know which cells contains boulders.

Fortunately, there always exists a path from the starting point to the exit.

Input Specification:

The first line contains a number, $T \leq 8$, which denotes the number of test cases that follow. Each test case has one integer, K, on a line by itself, denoting the number of boulders in the cells.

Output Specification:

For each test case, provide a sequence of moves, no longer than 1000, such that you are guaranteed to exit the grid from the starting point. Once you exit the grid, all subsequent moves are ignored. Print your sequence on a single line using only the characters "`NESW`", to denote North, East, South, West respectively.

You do not like to make useless moves, however, so the output string must become invalid if even a single character is removed from the end.

Sample Input:

```
1
0
```

Sample Output:

```
EESWWSEENNWWSE
```

Problem F- Monkey's Revenge

Suppose we are provided with a monkey that randomly presses a key on the keyboard one at a time. Given a phrase of length K, how many different randomly typed N-character strings are there which contain that phrase, as a substring? Write a program that outputs this result modulo 100003.

Assume that the keyboard and the phrase is made up only of lowercase alphanumeric characters. That is, `a` to `z` and `0` to `9`.

Input Specification:

The first line is an integer $T \leq 100$, the number of test cases, followed by $2T$ lines. The first line of each test case is the positive integer $N \leq 32$. The second line is the nonempty phrase of length $K \leq N$.

Output Specification:

Output the number of ways, modulo 100003.

Sample Input:

```
1
5
abcde
```

Sample Output:

```
1
```