Buy low, sell high. That is what one should do to make profit in the stock market (we will ignore short selling here). Of course, no one can tell the price of a stock in the future, so it is difficult to know exactly when to buy and sell and how much profit one can make by repeatedly buying and selling a stock.

But if you do have the history of price of a stock for the last n days, it is certainly possible to determine the maximum profit that could have been made. Instead, we are interested in finding the k_1 lowest prices and k_2 highest prices in the history.

Input

The input consists of a number of cases. The first line of each case starts with positive integers n, k_1, and k_2 on a line ($n \leq 1,000,000$, $k_1 + k_2 \leq n$, $k_1, k_2 \leq 100$). The next line contains integers giving the prices of a stock in the last n days: the i-th integer ($1 \leq i \leq n$) gives the stock price on day i. The stock prices are non-negative. The input is terminated by $n = k_1 = k_2 = 0$, and that case should not be processed.

Output

For each case, produce three lines of output. The first line contains the case number (starting from 1) on one line. The second line specifies the days on which the k_1 lowest stock prices occur. The days are sorted in ascending order. The third line specifies the days on which the k_2 highest stock prices occur, and the days sorted in descending order. The entries in each list should be separated by a single space. If there are multiple correct lists for the lowest prices, choose the lexicographically smallest list. If there are multiple correct lists for the highest prices, choose the lexicographically largest list.

Sample Input

```
10 3 2
1 2 3 4 5 6 7 8 9 10
10 3 2
10 9 8 7 6 5 4 3 2 1
0 0 0
```

Sample Output

```
Case 1
1 2 3
10 9
Case 2
8 9 10
2 1
```