D. Two out of Three

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya has recently developed a new algorithm to optimize the reception of customer flow and he considered the following problem.

Let the queue to the cashier contain n people, at that each of them is characterized by a positive integer a_i — that is the time needed to work with this customer. What is special about this very cashier is that it can serve two customers simultaneously. However, if two customers need a_i and a_j of time to be served, the time needed to work with both of them customers is equal to $max(a_i, a_j)$. Please note that working with customers is an uninterruptable process, and therefore, if two people simultaneously come to the cashier, it means that they begin to be served simultaneously, and will both finish simultaneously (it is possible that one of them will have to wait).

Vasya used in his algorithm an ingenious heuristic — as long as the queue has more than one person waiting, then *som e two people of the first three standing in front of the queue are sent simultaneously*. If the queue has only one customer number i, then he goes to the cashier, and is served within a_i of time. Note that the total number of phases of serving a customer will always be equal to $\lceil n / 2 \rceil$.

Vasya thinks that this method will help to cope with the queues we all hate. That's why he asked you to work out a program that will determine the minimum time during which the whole queue will be served using this algorithm.

Input

The first line of the input file contains a single number n ($1 \le n \le 1000$), which is the number of people in the sequence. The second line contains space-separated integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^6$). The people are numbered starting from the cashier to the end of the queue.

Output

Print on the first line a single number — the minimum time needed to process all n people. Then on $\lceil n / 2 \rceil$ lines print the order in which customers will be served. Each line (probably, except for the last one) must contain two numbers separated by a space — the numbers of customers who will be served at the current stage of processing. If n is odd, then the last line must contain a single number — the number of the last served customer in the queue. The customers are numbered starting from 1.

Examples

input
4 1 2 3 4
output
6 1 2 3 4

input
5 2 4 3 1 4
output
8 1 3 2 5 4