1 Problem

We play a game where we are given a sequence of n balloons, neatly placed from left to right. Each balloon is assigned a fixed integer value, which is printed on the balloon. We are given a needle to pop the balloons. When we pop a balloon, we earn a reward equal to the value of the popped balloon multiplied by the values of the two adjacent balloons. We are not allowed to pop the first and last balloons in the sequence, hence any popped balloon necessarily has two neighbors.

Unfortunately, the balloons in this game are a little delicate. When a balloon is popped with a needle, its loud noise pops the two adjacent balloons with it. Luckily, the balloons popped by the noise do not themselves make much noise, thus, they do not trigger further pops. When the balloons get popped (either directly with a needle or indirectly with noise), they are removed from the sequence for good.

The game finishes when there are no balloons left. To make sure you can complete the game under the given conditions, the number of balloons is always given as a multiple of 3. (Think what would happen otherwise.)

Write a program that finds the maximum total reward achievable in this game. Your program should also print the popping order that yields this reward. (If there are multiple orders, printing any is sufficient.) When specifying a balloon to be popped, your program should mention its position (from the left) in the original sequence of balloons, which is used as a fixed identifier for the balloon.

2 Input Format

```
N
V_1 V_2 ... V_N
```

`N` = The number of balloons in the initial sequence.
$\texttt{V}_i$ = The value of the ith balloon from the left in the initial sequence.

3 Output Format

```
R
P_1 P_2 ... P_{N/3}
```

`R` = The maximum total reward achievable in the game.
$\texttt{P}_i$ = The original position (as a 1-based index from the left) of the balloon to pop at the ith step.

4 Limits

$3 \leq \texttt{N} \leq 240$
$1 \leq \texttt{V}_i \leq 100$
Time Limit: 1 second
Memory Limit: 256 MB

5 Clarifications

- A well-written solution with $O(N^5)$ time is expected to get full points in this THE. (Though, there also exists an $O(N^3)$ solution.) If you are unable to code any such solution, please code a slower one for potential partial points.

- Your solution is expected as a C++ program source named `delicate.cpp` that reads from the standard input and writes to the standard output.
- It is OK to copy code from the sample codes we shared in our course website in ODTÜClass for this THE.
- Submit your code via ODTÜClass. You can use the evaluate functionality to auto-grade your solution.
- The grade from the auto-grader is not final. We can later do further evaluations of your code and adjust your grade. Solutions that do not attempt a "reasonable solution" to the given task may lose points.
- We will compile your code on g++ with options: `-std=c++17 -O2 -lm -Wall -Wextra -Wpedantic`

6 Hints

- A dynamic programming algorithm that solves $O(N^2)$ subproblems (each of which costs $O(N^3)$ to solve) achieves the time bound mentioned above.
- The size of the input is a multiple of 3. Maybe you should stick to that when thinking about subproblems.
- In case your memory is affecting your thinking: This problem is <u>not</u> circular and it does <u>not</u> require dealing with dynamic positions.

7 Examples

Sample Input 1

```
9
6 4 4 4 8 2 4 6 2
```

Sample Output 1

```
232
4 6 8
# Balloon Ids:  1  2  3  4  5  6  7  8  9
# Values:       6  4  4  4  8  2  4  6  2
# Balloon #4 is popped. Brings a reward of 4 * 4 * 8 = 128.
# Balloon Ids:  1  2  6  7  8  9
# Values     :  6  4  2  4  6  2
# Balloon #6 is popped. Brings a reward of 4 * 2 * 4 = 32.
# Balloon Ids:  1  8  9
# Values     :  6  6  2
# Balloon #8 is popped. Brings a reward of 6 * 6 * 2 = 72.
# No balloons left.
```

Sample Input 2

```
12
6 5 8 1 7 3 1 1 4 3 2 5
```

Sample Output 2

```
337
2 7 9 11
# Balloon Ids:  1  2  3  4  5  6  7  8  9 10 11 12
```

```
# Values:       6  5  8  1  7  3  1  1  4  3  2  5
# Balloon #2 is popped. Brings a reward of 6 * 5 * 8 = 240.
# Balloon Ids:  4  5  6  7  8  9 10 11 12
# Values     :  1  7  3  1  1  4  3  2  5
# Balloon #7 is popped. Brings a reward of 3 * 1 * 1 = 3.
# Balloon Ids:  4  5  9 10 11 12
# Values     :  1  7  4  3  2  5
# Balloon #9 is popped. Brings a reward of 7 * 4 * 3 = 84.
# Balloon Ids:  4 11 12
# Values     :  1  2  5
# Balloon #11 is popped. Brings a reward of 1 * 2 * 5 = 10.
# No balloons left.
```

Sample Input 3

```
15
6 2 4 3 6 1 1 6 9 8 1 1 7 2 3
```

Sample Output 3

```
579
3 9 7 13 5
# Balloon Ids:  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
# Values:       6  2  4  3  6  1  1  6  9  8  1  1  7  2  3
# Balloon #3 is popped. Brings a reward of 2 * 4 * 3 = 24.
# Balloon Ids:  1  5  6  7  8  9 10 11 12 13 14 15
# Values     :  6  6  1  1  6  9  8  1  1  7  2  3
# Balloon #9 is popped. Brings a reward of 6 * 9 * 8 = 432.
# Balloon Ids:  1  5  6  7 11 12 13 14 15
# Values     :  6  6  1  1  1  1  7  2  3
# Balloon #7 is popped. Brings a reward of 1 * 1 * 1 = 1.
# Balloon Ids:  1  5 12 13 14 15
# Values     :  6  6  1  7  2  3
# Balloon #13 is popped. Brings a reward of 1 * 7 * 2 = 14.
# Balloon Ids:  1  5 15
# Values     :  6  6  3
# Balloon #5 is popped. Brings a reward of 6 * 6 * 3 = 108.
# No balloons left.
```