**Algorithms with Java: Exam 28-06-2020**

This document defines the exam for ["Algorithms – Fundamentals (Java)" course @ Software University](https://softuni.bg/trainings/2991/algorithms-fundamentals-with-java-may-2020). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2484/Algorithms-Fundamentals-with-Java-Exam-28-June-2020).

# 2. Nuclear Waste

*The way it works is simple any fuel we use to produce energy has some need of disposal solution, for the unused waste. For example we think it is a "good idea" to dispose the carbon dioxide straight in to the atmosphere, when burning coal or any other fossil fuel.*

It is a bit different when a nuclear power plant needs to replace the fuel inside a reactor or the cooling rods, the old one become waste. They are highly radioactive and require much more complex process which often includes transporting the radioactive materials to disposal facilities often far away from the power plants and somewhere deep underground.

Still business is business and you are looking for the cheapest possible way to transport the nuclear waste to some of the facilities. The transport company offers to transport **flasks** of waste at **different** **cost** for the different **count** of **flasks**. You will be given this information as a sequence of integers as an example:

**12 20 30 40** meansthat **1** flasktransport **costs 12, two** costs **20, three** costs **30** etc**.**

You will also be given **N** the **number** of **flasks** you want to transport.

## Input

The input will come from the console on two lines.

* On the first line the **sequence** representing the **cost** of the **flasks** transport.
* On the second line single integer **N** the number of flasks you need to transport.

## Output

* First print the **minimum** cost for **all the** flasks transport. In the format: **"Cost: {minimumCost}"**
* Then print the optima setup as the **count** **of** **flasks** and the **corresponding** **cost**.
* **"{flasksCount} => {cost}"**
* **Note**: the **flasks** should be **printed** in **increasing** **order** by **count** of **flasks** transported **NOT** by cost.

## Constraints

* All input lines will be **valid integers you do not need to check that.**
* The range of the integers will be **[1…10000]**

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 12 21 31 43 79  10 | Cost: 104  2 => 21  2 => 21  3 => 31  3 => 31 |
| 12 21 31 40 49 58 69 79 90 101  15 | Cost: 147  3 => 31  6 => 58  6 => 58 |

*“Wisdom comes from experience. Experience is often a result of lack of wisdom.”*

*― Terry Pratchett*