I: Water troubles

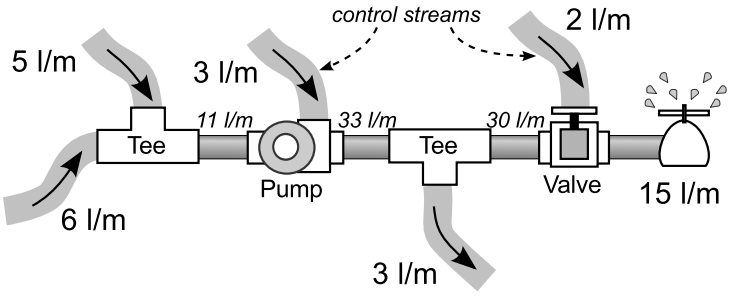
Source file name: `water.c, water.cpp`, `water.java`, *or* `water.py`
Author: M. Sánchez

Because of climate change, the little town of Minca in the Sierra Nevada is facing serious problems with its water supplies. They were used to get water from the abundant streams, brooks, and small rivers coming down the mountain, but now all of these carry a lot less water. On top of that, their coffee fields depend on irrigation machines and require precise amounts of water in order to survive.

Some ingenious hydraulic engineers that heard about Minca's problems decided to pay them a visit to help them save their delicate crops. They brought with them a series of gadgets and pipe fittings to divert water streams and gather just the right amount of water that coffee requires. None of those devices depend on electricity or fuel, making them perfect for usage in the mountain. The first kind of device was just a tee pipe: depending on the way it is installed, it combines two streams or divides a stream in two parts (which may transport different amounts). The second kind of device was a water controlled water pump: it takes a controlling stream of water on one of its input fittings and augments the other input stream by a factor that depends on the controlling stream. Finally, there were hydraulic valves that reduce input streams by a factor that also depends on some controlling water stream.

Everything was joy in Minca because these ingenious devices were going to help the town properly operate its irrigation machines. Unfortunately, the engineers quickly discovered that, in order to send the proper amount of water to each field, they would have to combine in creative ways their machines and the flexible and durable hoses that they use to take water from the streams and to dispose water on them when unused.

In the following figure you can see how they solved the problem of getting 15 liters of water per minute to some field, considering that they had with them hoses for 6, 3, 2, 5, 20 and 3 l/m. The first tee pipe receives 5 l/m and 6 l/m from the corresponding hoses. Next, a hydraulic pump controlled by a 3 l/m stream feeds 33 l/m to the next tee pipe, which discards 3 l/m. Finally, the valve is controlled by a 2 l/m stream, taking down its output to the 15 l/m that the field and its irrigation system requires. The 20 l/m hose was not used. It were used the hoses for 5, 6, 3, 3 and 2 l/m, in that order.



Your task is to help Mincans and the hydraulic engineers to assemble their devices and hoses in linear sequences to save the fields. For this, you will receive the required amount of water required by a given field, and the capacity of the hoses available to grab and dispose water from the brooks and small rivers.

For simplicity, assume that engineers have an unlimited amount of every type of device, that there is a single irrigation machine in the coffee field, that a tee device that divides a stream always sends the extra water back to a brook, and that the water flows must always be integers. Also, you do not have to consider the pipes and tubes used to connect the devices between them, or to connect the devices to the irrigation machine.

Input

The input consists of several test cases. Each test case is described by a single line containing $H + 1$ blank-separated integers $T, c_1, c_2, \ldots, c_H$, where T indicates the amount of water required by a certain field in the town ($1 \leq T \leq 10^{15}$), H is the number of hoses ($1 \leq H \leq 7$), and $c_1, c_2, \ldots, c_H$ indicate the capacity of the hoses available to direct the water ($1 \leq c_i \leq 50$ for each $1 \leq i \leq H$).

The last test case is followed by a line containing a single zero value.

The input must be read from standard input.

Output

For each test case, print a line with a single integer. This integer should indicate the amount of water that may be sent into the field with an optimal arrangement of the hoses and the devices. If it is impossible to find an arrangement that sends the exact quantity of water required, your program should find the closest quantity that is superior to the requirement, or 0 if it is impossible to reach the required amount.

The output must be written to standard output.

Sample Input	**Sample Output**
15 6 3 2 5 20 3 15 20 5 6 3 3 2 15 5 6 3 3 2 10 1 2 3 8 6 4 5 14 4 3 7 5 11 3 3 3 6 3 3 0	15 15 15 0 9 14 12 6