



## Detecting Molecules

Petr is working for a company that has built a machine for detecting molecules. Each molecule has an integer weight. The machine has a *detection range*  $[l, u]$ , where  $l$  and  $u$  are integer. The machine can detect a set of molecules if and only if this set contains a subset of the molecules with total weight belonging to the machine's detection range.

Formally, consider  $n$  molecules with positive integer weights  $w_0, \dots, w_{n-1}$ . The detection is successful if there is a set of distinct indices  $I = i_1, \dots, i_m$  such that  $l \leq w_{i_1} + \dots + w_{i_m} \leq u$ .

Due to specifics of the machine, the gap between  $l$  and  $u$  is guaranteed to be greater than or equal to the weight gap between the heaviest and the lightest molecule. Formally,  $u - l \geq w_{max} - w_{min}$ , where  $w_{max} = \max(w_0, \dots, w_{n-1})$  and  $w_{min} = \min(w_0, \dots, w_{n-1})$ .

Your task is to write a program which either finds any one subset of molecules with total weight within the detection range, or determines that there is no such subset.

### Implementation details

You should implement one function (method):

- `int[] solve(int l, int u, int[] w)`
  - $l$  and  $u$ : the endpoints of the detection range,
  - $w$ : weights of the molecules.
  - if the required subset exists, the function should return an array of indices of molecules that form any one such subset. If there are several correct answers, return any of them.
  - if the required subset does not exist, the function should return an empty array.

For the C language the function signature is slightly different:

- `int solve(int l, int u, int[] w, int n, int[] result)`
  - $n$ : the number of elements in  $w$  (i.e., the number of molecules),
  - the other parameters are the same as above.
  - instead of returning an array of  $m$  indices (as above), the function should write the indices to the first  $m$  cells of array `result` and then return  $m$ .
  - if the required subset does not exist, the function should not write anything to the `result` array and it should return `0`.

Please use the provided template files for details of implementation in your programming language.

## Examples

### Example 1

`solve(15, 17, [6, 8, 8, 7])`

In this example we have four molecules with weights 6, 8, 8 and 7. The machine can detect subsets of molecules with total weight between 15 and 17, inclusive. Note, that  $17 - 15 \geq 8 - 6$ . The total weight of molecules 1 and 3 is  $w_1 + w_3 = 8 + 7 = 15$ , so the function can return `[1, 3]`. Other possible correct answers are `[1, 2]` ( $w_1 + w_2 = 8 + 8 = 16$ ) and `[2, 3]` ( $w_2 + w_3 = 8 + 7 = 15$ ).

### Example 2

`solve(14, 15, [5, 5, 6, 6])`

In this example we have four molecules with weights 5, 5, 6 and 6, and we are looking for a subset of them with total weight between 14 and 15, inclusive. Again, note that  $15 - 14 \geq 6 - 5$ . There is no subset of molecules with total weight between 14 and 15 so the function should return an empty array.

### Example 3

`solve(10, 20, [15, 17, 16, 18])`

In this example we have four molecules with weights 15, 17, 16 and 18, and we are looking for a subset of them with total weight between 10 and 20, inclusive. Again, note that  $20 - 10 \geq 18 - 15$ . Any subset consisting of exactly one element satisfies the requirement, so correct answers are: `[0]`, `[1]`, `[2]` and `[3]`.

## Subtasks

- (9 points):  $n \leq 100$ ,  $w_i \leq 100$ , all  $w_i$  are equal.
- (10 points):  $n \leq 100$ ,  $w_i \leq 1000$ , and  $\max(w_0, \dots, w_{n-1}) - \min(w_0, \dots, w_{n-1}) \leq 1$ .
- (12 points):  $n \leq 100$  and  $w_i, u, l \leq 1000$ .
- (15 points):  $n \leq 10000$  and  $w_i, u, l \leq 10000$ .
- (23 points):  $n \leq 10000$  and  $w_i, u, l \leq 500000$ .
- (31 points):  $n \leq 200000$  and  $w_i, u, l < 2^{31}$ .

## Sample grader

The sample grader reads the input in the following format:

- line 1: integers  $n$ ,  $l$ ,  $u$ .
- line 2:  $n$  integers:  $w_0, \dots, w_{n-1}$ .