# Distributing Chocolates

There are $k$ bags numbered from 0 to $k-1$ with the $i$th bag containing $n_i$ chocolates. The total number of chocolates in all bags is $n = \sum_{0 \le i < k} n_i$. It is required to redistribute the chocolates so that there are nearly equal number of chocolates in each bag, that is the difference must be at most 1. However, the only way allowed for doing this is the following. Pick all chocolates from any one bag, say the $i$th, and redistribute the $n_i$ chocolates in the bag equally among all the bags. This is done by placing one chocolate in bag $(i+1) \bmod k$ and then repeatedly moving to the next bag, placing one chocolate in each, continuing with bag 0 after bag $k-1$, till all $n_i$ chocolates have been placed in bags. Note that some of the chocolates may be placed back in bag $i$ itself. Thus if there were 4 bags containing 6, 3, 8, 5 chocolates, if we redistribute bag 1, the bags would contain 7, 0, 9, 6 chocolates, and if bag 3 is redistributed, the bags would contain 8, 4, 9, 1 chocolates. Given the initial distribution of chocolates in the bags, you have to write a program to find a sequence of steps that will achieve a nearly equal redistribution. Note that any valid sequence is okay, it does not have to be the shortest.

**Input/Output Format:** The first line of input will specify two positive integers $n$ and $k$ in that order. The next line will contain $k$ non-negative numbers separated by a space, giving the $n_i$ values. The value of $n$ will be at most 100000 and the value of $k$ at most 100. The output must be a sequence of steps that achieves the redistribution. Each step is specified by the number of the bag whose contents are redistributed. Successive numbers must be separated by a space and all printed on the same line. If no moves are required, print nothing. A file called check.cpp is put up on Moodle to check your output. If you compile it as g++ -o check check.cpp and then run ./check infile outfile, it will output 'Correct' if the output in file outfile is correct for the input in file infile. For compiling, you can use g++ -O2 file_name, where O2 indicates the optimization level. This will reduce the execution time, and can also detect some errors that normal compilation may not. There are more efficient data structures possible for this problem, but for this lab, implementation using just an array would suffice. The time required should be at most 3 seconds, with optimization.

**Sample Input/Output**

| Input | Possible output |
|---|---|
| 12 3<br>3 5 4 | 2 1 1 2 2 1 2 0 |

**Hint:** If we get all chocolates in one bag, then it is easy to distribute nearly equally. One way of ensuring that a sequence of steps ends up with a given distribution is to define an order relation on the distributions, such that the required distribution is the minimum element. At every step ensure that you move to a distribution that is 'smaller' in the ordering. Since there are only finitely many distributions, you will eventually reach the minimum. Can you think of such a relation for this problem?

**Submission:** Name the file as RollNo_2.cpp and submit on moodle.

**Homework:** A more general problem is given arbitrary initial and final distributions of chocolates, find a sequence of steps that leads from the initial to the final. Is this always possible? What can you say about the number of moves required?