

7. Unpacking (pakk)

10 sec

100 points

You are given a set of files. Your goal is to compress these files as compactly as possible.

Input. There are 10 input files, which you can download from CMS. (Note that all inputs use Unix-style newlines.)

Output. For each input file, submit a program that prints the corresponding input file to standard output.

Due to limitations in CMS, each program must be submitted as a `.txt` file. The used programming language must thus be marked in a comment on the first line of the file. Depending on the used language, the first line should be one of the following:

Language	First line	Compilation command
C23	<code>// LANGUAGE: C</code>	<code>gcc -std=gnu23 -O2 input.c</code>
C++23	<code>// LANGUAGE: C++</code>	<code>g++ -std=gnu++23 -O2 input.cpp</code>
Python 3.11 (PyPy)	<code># LANGUAGE: PyPy</code>	<code>pypy3 input.py</code>

You are allowed to use different programming languages for different inputs.

Remark. Submitted solutions will be compiled and executed in a testing environment different from the usual one. This environment does not include any standard compression utilities or libraries. A copy of this environment in the form of a Docker container is available at ghcr.io/e-i-o/pakk-runtime. (The environment also has no shell; you may use the `pypy3` REPL in its place.)

In this task, solutions are allowed to use up to 1 GB of RAM instead of the usual 256 MB.

Grading. Each input is worth 10 points. The *length* of a program is considered to be its length in bytes, excluding the first “LANGUAGE:” line. Let ℓ be the length of the submitted program and b be the length of the shortest submitted correct program for the corresponding input (over all contestants). A (correct) program will get $10 \cdot (b/\ell)^3$ points for the corresponding test case.

Example.	Input	Output (submitted program)
	Hello	<code># LANGUAGE: PyPy</code>
	Hello	<code>print(end="Hello\n"*4)</code>
	Hello	
	Hello	

The length of this program would be 22 bytes. (Note that in this this example, the input has a trailing newline, but the output does not. The inputs may or may not have terminating newlines; the program must output the exact same input down to the byte. The submitted program itself does not need to have a newline, though some text editors might add one by default.)

Hints

- `input_003.txt`: This input is a linear combination of well-known real numbers.
- `input_004.txt`: This input is a sequence (x_i) , where $x_i = \sum_{j=1}^k a_j x_{i-j} \pmod{10^9 + 7}$ for some natural numbers k and $a_1, \dots, a_k$.
- `input_005.txt`: This input is a sequence (x_i) , where $x_i = \sum_{j=1}^k a_j x_{i-j} \pmod{n}$ for some natural numbers n, k and $a_1, \dots, a_k$.