UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

The Coding DEI

Paolo Pellizzoni, Matteo Zappia,
Simone Moretti and Luca Dolci

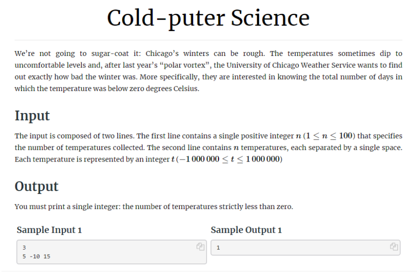
University of Padova, Italy

Introduction

What is competitive programming?

Given well-known Computer Science problems, solve them as quickly as possible!

- no "open" problems
- exact solutions only
- speed is a factor



Cold-puter Science

We're not going to sugar-coat it: Chicago's winters can be rough. The temperatures sometimes dip to uncomfortable levels and, after last year's "polar vortex", the University of Chicago Weather Service wants to find out exactly how bad the winter was. More specifically, they are interested in knowing the total number of days in which the temperature was below zero degrees Celsius.

Input

The input is composed of two lines. The first line contains a single positive integer n ($1 \leq n \leq 100$) that specifies the number of temperatures collected. The second line contains n temperatures, each separated by a single space. Each temperature is represented by an integer t ($-1\,000\,000 \leq t \leq 1\,000\,000$)

Output

You must print a single integer: the number of temperatures strictly less than zero.

Sample Input 1

```
3
5 -10 15
```

Sample Output 1

```
1
```

Introduction

Why competitive programming?

- become better at problem solving (exams, research, real-life)
- train for coding interviews (Google, Facebook, Bending Spoons)
- add stuff to your CV
- win prizes and travel to cool places

Introduction

Popular coding competitions

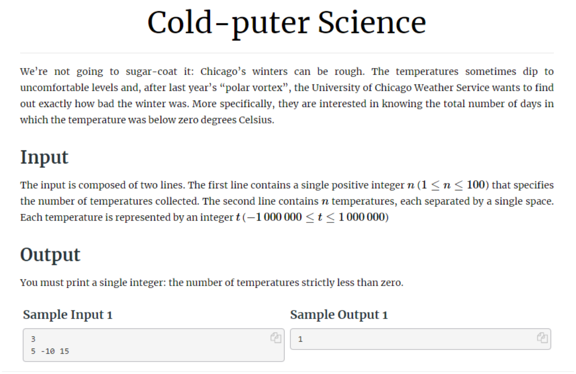
- International Olympiad in Informatics (for high school students)
- ACM International Collegiate Programming Constest (for college students, SWERC is one of ICPC competitions)
- Google Code Jam, open to anyone and very challenging
- FB hacker Cup, similar to GCJ

Introduction

Other types of coding competitions

- Optimization problems: no feasible exact solution, find heuristics for solving the problem (Google Hash Code, Reply Challenge)
- Hackathons: solve some open ended problem, often related to real-life needs
- Cybersecurity: solve well-known cybersecurity problems

Back to our example



Cold-puter Science

We're not going to sugar-coat it: Chicago's winters can be rough. The temperatures sometimes dip to uncomfortable levels and, after last year's "polar vortex", the University of Chicago Weather Service wants to find out exactly how bad the winter was. More specifically, they are interested in knowing the total number of days in which the temperature was below zero degrees Celsius.

Input

The input is composed of two lines. The first line contains a single positive integer n ($1 \leq n \leq 100$) that specifies the number of temperatures collected. The second line contains n temperatures, each separated by a single space. Each temperature is represented by an integer t ($-1\,000\,000 \leq t \leq 1\,000\,000$)

Output

You must print a single integer: the number of temperatures strictly less than zero.

Sample Input 1

```
3
5 -10 15
```

Sample Output 1

```
1
```

Possible outcomes: AC, WA, TLE, RE, MLE, ...

Useful resources

Books

- Halim - Competitive Programming 3
- Laaksonen - Guide to Competitive Programming
- Cormen - Introduction to Algorithms

Sites

- Kattis
- Codeforces
- UVa Online Judge
- Google Code Jam/Kick Start
- FB Hacker Cup

Important prerequisites

Identifying quickly problem types:

- Ad hoc
- Complete search
- Greedy
- DP
- Graphs
- Mathematics
- Strings
- Computational Geometry

Important prerequisites

Complexity analysis:
All problems have a time limit (e.g. 1 second) and the solution must run within the given time.
Modern computers can process up to $100M$ (10^8) operations in a few seconds.
Example: $N = 10^5$

- $\mathcal{O}(n^2) \simeq 10^{10}$: TLE
- $\mathcal{O}(n \log n) \simeq 5 * 10^6$: AC
- $\mathcal{O}(n\sqrt{n}) \simeq 3 * 10^7$: AC if you write optimized code

It may be useful to include constants in the notation, e.g. $O(100 * n \log n)$

Important prerequisites

Complexity analysis:

- $\mathcal{O}(n!)$: $n <$
- $\mathcal{O}(2^n)$: $n <$
- $\mathcal{O}(n^4)$: $n <$
- $\mathcal{O}(n^3)$: $n <$
- $\mathcal{O}(n^2)$: $n <$
- $\mathcal{O}(n \log n)$: $n <$
- $\mathcal{O}(n)$: $n <$
- $\mathcal{O}(\log n)$: $n <$

Important prerequisites

Complexity analysis:

- $\mathcal{O}(n!)$: $n < 10$
- $\mathcal{O}(2^n)$: $n < 25$
- $\mathcal{O}(n^4)$: $n < 100$
- $\mathcal{O}(n^3)$: $n < 400$
- $\mathcal{O}(n^2)$: $n < 5000$
- $\mathcal{O}(n \log n)$: $n < 10^5$
- $\mathcal{O}(n)$: $n < 5 * 10^6$
- $\mathcal{O}(\log n)$: $n < 10^{18}$

Important prerequisites

Know your language:

Different languages have different applications:

- C++: very fast, probably the best for competitive programming.
  Cons: less warnings, more difficult to debug, no bigint
- Java: quite fast and versatile, easier to debug. Has builtin BigInt.
  Cons: very verbose, slower than c++
- Python: extremely versatile. Has builtin BigInt. *Cons:* quite slow, more difficult to debug.

Whatever language you use, it's important to know its features, such as the standard libraries.

Dynamic Arrays

C++ vector:

- included in `bits/stdc++.h`
- resizable array, use `push_back`
- sort with `sort(v.begin(), v.end())`

Java ArrayList:

- included in `java.utils`
- resizable array, use `.add()`
- sort with `Collections.sort()`

Dynamic Arrays

Complexity:

- Access $\rightarrow \mathcal{O}(1)$
- One side insertion $\rightarrow \mathcal{O}(1)$
- Deletion $\rightarrow \mathcal{O}(n)$
- Search $\rightarrow \mathcal{O}(n)$

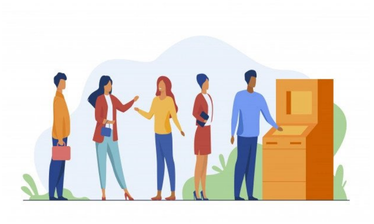
https://open.kattis.com/problems/lineup

Queue and Stack

- Queue implemented FIFO methods (First Input First Output)
- Stack implemented LIFO methods (Last Input First Output)
- Both have insertion and deletion in $\mathcal{O}(1)$

Method	Queue C++	Time	Stack C++	Time
Insertion	.push()	$\mathcal{O}(1)$	.push()	$\mathcal{O}(1)$
Deletion	.pop()	$\mathcal{O}(1)$	.pop()	$\mathcal{O}(1)$
Access(first element)	.front()	$\mathcal{O}(1)$	.top()	$\mathcal{O}(1)$

Queue and Stack



Priority Queue

- C++: `priority_queue`, Java: `PriorityQueue`
- C++: the first element is the greatest of all elements in the queue, Java: first element is the smallest
- It's a binary heap

Method	Priority queue C++	Time
Insertion	.push()	$\mathcal{O}(\log(n))$
Deletion	.pop()	$\mathcal{O}(\log(n))$
Access(first element)	.top()	$\mathcal{O}(1)$

Maps and Sets

- C++: `set/map`, Java: `TreeSet/TreeMap`
- Sets are containers that store unique elements following a specific order
- Maps are containers that store elements formed by a combination of a key value and a mapped value, following a specific order
- Both sets and maps are implemented as binary search trees

Method	Sets C++	Time	Maps C++	Time
Insertion	.insert()	$\mathcal{O}(log(n))$	.insert()	$\mathcal{O}(log(n))$
Deletion	.erase()	$\mathcal{O}(log(n))$	.erase()	$\mathcal{O}(log(n))$
Access	[unique_key]	$\mathcal{O}(log(n))$	"	"
Search	.find()	$\mathcal{O}(log(n))$	.find()	$\mathcal{O}(log(n))$

Hash Maps and Sets

- C++: `unordered_set/unordered_map`, Java: `HashSet/HashMap`
- They allow constant time in all insertion deletion access and search

https://open.kattis.com/problems/competitivearcadebasketball
https://open.kattis.com/problems/doctorkattis

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione