icpc international collegiate programming contest

ICPC Europe Contests

Central Europe Regional Contest

Practice Session

2022
2023

europe

icpc.foundation

JET BRAINS

icpc global sponsor
programming tools

HUAWEI

icpc diamond
multi-regional sponsor

X – Chronogram

Time limit: 1 s *Memory limit: 256 MiB*

A *chronogram* is an inscription (usually found on monuments) in which the year of the event commemorated by the monument is encoded. The year is obtained by adding up all the values of those letters that can appear in Roman numerals. Values of individual letters:

- `I` ... 1
- `V` ... 5
- `X` ... 10
- `L` ... 50
- `C` ... 100
- `D` ... 500
- `M` ... 1000

Chronograms are usually studied by theologians. As a part of a study of great importance in the Central Europe, they compiled a long list of chronograms. The monks would like to make sure they do not make any mistake in calculating the corresponding years and therefore decided to use a computer program. Since monks are not skilled programmers, they urgently need your help.

Write a program that will extract the year of the event from a chronogram.

Input data

The first line of the input contains an integer T, i.e. the number of test cases. This is followed by T lines; each line contains one chronogram that contains at most 1000 characters. The text contains only uppercase letters of the English alphabet and spaces.

Input limits

- $1 \leq T \leq 10000$
- Every line contains at most 1000 characters.
- A character may be an uppercase letter of the English alphabet or a space.

Output data

For each chronogram output one integer, i.e. the year encoded in the chronogram.

Example

Input

```
2
VIDE ISTA ECCLESIA CATHEDRALIS SVB PRAESVLE FRANCISCO PAROCHOQVE STANSLAO
                                                 LAETA EXSISTIT REVIVISCENS
ENTERTAINING REGIONAL CONTEST HELD IN EXTRAVAGANT LJUBLJANA WITH
                                                    RIDICULOUSLY HARD TASKS
```

Output

```
1999
2022
```

Note. The input contains two lines with chronograms. Since the two chronogram do not fit on an A4 page, line-breaks were used. The actual input data does not contain these line-breaks.

Y – Permutations

Time limit: 2 s   Memory limit: 256 MiB

Let $[n]$ denote the set $\{1, 2, \dots, n\}$. A **permutation** of order n is a one-to-one mapping from the set $[n]$ to the set $[n]$. For example, let $\pi\colon [7] \to [7]$ be a concrete permutation of order 7. We usually represent it with a table like this:

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 7 & 4 & 1 & 6 & 5 & 2 \end{pmatrix}.$$

This means that $\pi(1) = 3$, $\pi(2) = 7$, $\pi(3) = 4$ etc. Since the top row is always $1\ 2\ 3 \dots n$, we usually omit it and write down the permutation in the **one-line notation**:

$$\pi = 3\ 7\ 4\ 1\ 6\ 5\ 2.$$

The **cycle notation** is also very popular. We start with element 1 and determine its image $\pi(1) = 3$. Next, we take the element 3 and determine its image $\pi(3) = 4$. If we keep doing this, we sooner or later arrive back at the element 1. Indeed, $\pi(4) = 1$. The permutation π contains the cycle $(1\ 3\ 4)$. Then we take the smallest number that has not yet appeared in any cycle – in our case it is 2 – and repeat the process. Eventually, we obtain to the following:

$$\pi = (1\ 3\ 4)\ (2\ 7)\ (5\ 6).$$

A permutation of order n can also be represented by an **inversion table** $b_1\ b_2\ b_3 \dots b_n$ ($0 \leq b_i \leq n - i$), where b_i is the number of those elements that are greater than i and appear to the left of i in the one-line notation. In our concrete example, the inversion table is:

$$3\ 5\ 0\ 1\ 2\ 1\ 0.$$

It is known that every permutation can be written in a unique way by an inversion table and that every inversion table $b_1\ b_2\ b_3 \dots b_n$ in which $0 \leq b_i \leq n - i$ holds for all $i \in [n]$, represents a valid permutation.

Write a program that will convert a inversion table to its corresponding cycle notation.

Input data

The input data consists of two lines. The first line contains an integer n, i.e. the order of a permutation. The second line contains n space-separated integers describing a valid inversion table.

Input limits

- $1 \leq n \leq 10^5$

Output data

Output one line that contains the cycle notation of the permutation given by the inversion table. Each cycle should be in parentheses. Cycles should be separated by one space. The numbers within the cycle should also be separated by one space. The smallest element should always be in the first place in the cycle. Cycles should be ordered according to the element in the first place. (See examples.)

Examples

Input	Output
7 3 5 0 1 2 1 0	(1 3 4) (2 7) (5 6)
6 4 2 1 1 0 0	(1 5) (2 3) (4) (6)

Z – Sokoban

Time limit: 10 s Memory limit: 512 MiB

If you are familiar with the Sokoban video game, the description of the task is quite short. You have to find the minimum number of box pushes to solve a given puzzle. The number of moves the player makes is not important.

The game Sokoban takes place on a grid, where in every step the player can move to one of the four adjacent squares in either the horizontal or vertical direction. Some squares are blocked by walls and some other squares are occupied by boxes. Certain squares are designated as storage locations. The player has to push boxes onto those storage locations. The player pushes a box by moving into its square, which pushes it into the square beyond. There must be a free square beyond the box, otherwise it cannot be moved. The player also cannot push boxes into other boxes or walls, and boxes cannot be pulled or transferred in any other way. The only allowed method of moving boxes is by pushing them. Even if the box is already on the storage location, it can still be moved. The only important thing is that all boxes end up on the storage locations in the end.

Even very small configurations can be extremely complex, so we need to use an efficient algorithm for solving Sokoban puzzles.

Input data

A grid of dimensions $N \times M$ is described in N lines with at most M characters. Some lines may contain less than M characters as they do not contain trailing spaces. It is guaranteed that the player is in a region that is enclosed with the walls. The number of boxes B equals the number of storage locations. The following list describes the meaning of each character that can be used to describe a puzzle:

```
'#' - a wall
' ' - free square
'.' - free storage location
'@' - the player standing on a square that is not a storage location
'+' - the player standing on a storage location
'$' - a box placed on a square that is not a storage location
'*' - a box placed on a storage location
```

Input limits

- $3 \le N, M \le 8$
- $1 \le B \le 4$

Output data

Output a single line that contains the minimum number of box pushes that are needed to solve the given puzzle. It is guaranteed that a puzzle is solvable.

Examples

Input

```
###
#.#####
#..   #
# $$$@#
#    ##
######
```

Output

```
9
```

Input

```
####
#. ####
#.$   #
#@ $$ #
###.  #
  #####
```

Output

```
13
```

Input

```
 ####
 #  #
##  ###
#.  $ #
# +$* #
#     #
#######
```

Output

```
6
```