Verifying the 8-Queens

Source file: queens.(java|c|cpp)
Input file: queens.in

The 8-Queens problem in chess is to place 8 queens on a chess board such that none of the queens is threatening any of the others. A chess board is an 8 x 8 matrix of squares and queens may move any direction along a row, column or diagonal. The problem is to input the 8 columns of the queens on the rows of a chess board, with 1 being the first column and 8 being the last, e.g. 1 2 3 4 5 6 7 8 means the queens are along the diagonal, which would not be a valid solution.

**Input Specification**

The input file consists of an unknown number of configurations where each configuration consists of a permutation of the numbers from 1 to 8 on consecutive lines.

**Output Specification**

If the configuration is valid, the following message should be printed:

*This is a valid configuration.*

Otherwise, this message should be printed:

*This is NOT a valid configuration.*

**Sample Input**

3 5 2 8 1 7 4 6
1 8 2 5 3 7 4 6

**Sample Output**

This is a valid configuration.
This is NOT a valid configuration.
Binomial Showdown

Source file: binomial.(java|c|cpp)
Input file: binomial.in

In how many ways can you choose $k$ elements out of $n$ elements, not taking order into account? Write a program to compute this number.

**Input Specification**

The input file will contain one or more test cases. Each test case consists of one line containing two integers $n$ ($n \geq 1$) and $k$ ($0 \leq k \leq n$). Input is terminated by two zeroes for $n$ and $k$.

**Output Specification**

For each test case, print one line containing the required number. This number will always fit into an integer, i.e. it will be less than $2^{31}$.

**Warning:** Don't underestimate the problem. The result will fit into an integer - but if all intermediate results arising during the computation will also fit into an integer depends on your algorithm. The test cases will go to the limit.

**Sample Input**

```
4  2
10 5
49 6
0  0
```

**Sample Output**

```
6
252
13983816
```
Compromise

Source file: compromise.(java|c|cpp)
Input file: compromise.in

In a few months the European Currency Union will become a reality. However, to join the club, the Maastricht criteria must be fulfilled, and this is not a trivial task for the countries (maybe except for Luxembourg). To enforce that Germany will fulfill the criteria, our government has so many wonderful options (raise taxes, sell stocks, revalue the gold reserves,...) that it is really hard to choose what to do.

Therefore the German government requires a program for the following task:
Two politicians each enter their proposal of what to do. The computer then outputs the longest common subsequence of words that occurs in both proposals. As you can see, this is a totally fair compromise (after all, a common sequence of words is something what both people have in mind).

Your country needs this program, so your job is to write it for us.

Input Specification

The input file will contain several test cases. Each test case consists of two texts. Each text is given as a sequence of lower-case words, separated by whitespace, but with no punctuation. Words will be less than 30 characters long. Both texts will contain less than 100 words and will be terminated by a line containing a single '#'. Input is terminated by end of file.

Output Specification

For each test case, print the longest common subsequence of words occurring in the two texts. If there is more than one such sequence, any one is acceptable. Separate the words by one blank. After the last word, output a newline character.
Sample Input

die einkommen der landwirte
sind fuer die abgeordneten ein buch mit sieben siegeln
um dem abzuwenden
muessen dringend alle subventionsgesetze verbessert werden
#
die steuern auf vermoegen und einkommen
sollten nach meinung der abgeordneten
nachdruecklich erhoben werden
dazu muessen die kontrollbefugnisse der finanzbehuerden
dringend verbessert werden
#

Sample Output

die einkommen der abgeordneten muessen dringend verbessert werden
Dungeon Master

Source file: dungeon.(java|c|cpp)
Input file: dungeon.in

You are trapped in a 3D dungeon and need to find the quickest way out! The dungeon is composed of unit cubes which may or may not be filled with rock. It takes one minute to move one unit north, south, east, west, up or down. You cannot move diagonally and the maze is surrounded by solid rock on all sides.

Is an escape possible? If yes, how long will it take?

Input Specification

The input file consists of a number of dungeons. Each dungeon description starts with a line containing three integers $L$, $R$ and $C$ (all limited to 30 in size).
$L$ is the number of levels making up the dungeon.
$R$ and $C$ are the number of rows and columns making up the plan of each level.
Then there will follow $L$ blocks of $R$ lines each containing $C$ characters. Each character describes one cell of the dungeon. A cell full of rock is indicated by a '#' and empty cells are represented by a '.'). Your starting position is indicated by 'S' and the exit by the letter 'E'. There's a single blank line after each level. Input is terminated by three zeroes for $L$, $R$ and $C$.

Output Specification

Each maze generates one line of output. If it is possible to reach the exit, print a line of the form
Escaped in $x$ minute(s).
where $x$ is replaced by the shortest time it takes to escape.
If it is not possible to escape, print the line
Trapped!
Sample Input

3 4 5
S....
.#.#.#
.#.#..
.#.#.#
#.#.#
#.#.#
#.#.#
.#.#.
#.#.
#.#
#.#
#.#
E

1 3 3
S..#
#E#
#

0 0 0

Sample Output

Escaped in 11 minute(s).
Trapped!
Addition Chains

An addition chain for $n$ is an integer sequence $<a_0, a_1, a_2, ..., a_m>$ with the following four properties:

- $a_0 = 1$
- $a_m = n$
- $a_0 < a_1 < a_2 < ... < a_{m-1} < a_m$
- For each $k$ ($1 \leq k \leq m$) there exist two (not necessarily different) integers $i$ and $j$ ($0 \leq i$, $j \leq k - 1$) with $a_k = a_i + a_j$

You are given an integer $n$. Your job is to construct an addition chain for $n$ with minimal length. If there is more than one such sequence, any one is acceptable. For example, $<1, 2, 3, 5>$ and $<1, 2, 4, 5>$ are both valid solutions when you are asked for an addition chain for 5.

Input Specification

The input file will contain one or more test cases. Each test case consists of one line containing one integer $n$ ($1 \leq n \leq 100$). Input is terminated by a value of zero (0) for $n$.

Output Specification

For each test case, print one line containing the required integer sequence. Separate the numbers by one blank.

Hint: The problem is a little time-critical, so use proper break conditions where necessary to reduce the search space.

Sample Input

5
7
12
15
77
0

Sample Output

1 2 4 5
1 2 4 6 7