South African Computer Olympiad Training DAY 1

Overview

Author	Bruce	Graham	Carl
Problem	**Jobs**	**Codes**	**Jugs**
Program name	`jobs.exe`	`codes.exe`	`jugs.exe`

Source name	`jobs.pas`
	`jobs.jav`
	`jobs.cpp`
		`codes.pas`
		`codes.jav`
		`codes.cpp`
			`jugs.pas`
			`jugs.jav`
			`jugs.cpp`

Input file	`stdin`	`stdin`	`stdin`
Output files(10)	`stdout`	`stdout`	`stdout`
Time limit	1 second	1 second	1 second
Num. of tests	10	10	10
Points per test	10	10	10
Total points	**100**	**100**	**100**

The maximum total score for Day1 is 300 points.

Scheduling jobs

Adapted from British Olympiad in Informatics 2001 By Bruce Merry

Description

Farmer John and his assistant are assembling a milking machine. Assembling the machine requires a number of steps, each of which takes exactly one hour (whether Farmer John or his assistant performs the step) and which only one person can work on. Once a step is started, it is worked on until it is finished. Farmer John wants to start milking his cows as soon as possible, so he is in a hurry to get the machine finished. While two people cannot work on the same step at the same time, they can work on two different steps. However they need to plan carefully, since some steps depend on other steps. Every step is depended on by at most one other step, but may depend on several previous steps. There are no cycles of dependencies.

Example

Suppose there are 6 steps that need to be performed, numbered 1 to 6. Step 5 depends on step 2, and step 6 depends on both steps 3 and 4. Once possible way to order the tasks is

Hour 1
    Steps 1, 2
Hour 2
    Steps 3, 5
Hour 3
    Step 4
Hour 4
    Step 6

However a better way to organise things is

Hour 1
    Steps 1, 2
Hour 2
    Steps 3, 4
Hour 3
    Steps 4, 5

Input file (jobs.in)

The first line of the input file contains *N*, the number of steps (numbered 1 to N). The next *N* lines each contain a single integer. The *i*th of these is the number of the step that depends on step *i*, or the value 0 if there is no step that depends on step *i*.

Sample input

```
6
0
5
6
6
0
0
```

Output format (jobs.out)

The first line of the output file contains *H*, the number of hours required to complete the assembly. The following *H* lines each contain two space separated integers. The *ith* of these lines contains the numbers of the steps performed during hour *i*. If only one step was performed during some hour, the second of these integers should be a 0 (at least one step must be performed every hour).

Sample output

```
3
1 2
3 4
5 6
```

Time limit

1 second.

Constraints

1 <= *N* <= 50000

Scoring

Any of the following scores 0.

- Not performing a step
- Performing a step twice
- Performing a non-existent step, other than using 0 as the second step on a line.
- Performing a step before all of the steps on which it depends.

An optimal solution scores 100%. Otherwise let the optimal solution be *K* hours and your solution be *L* hours. You score (*K* + *N* - 2*L*) / (*N* - *K*) * 100% rounded down, or zero if this would be negative.

Codes

Adapted from 2002 ACM-ICPC Final
By Graham Poulter

Description

Phil Oracle has a unique ability that makes him indispensable at the National Spying Agency. His colleagues can bring him any new binary code and he can tell them immediately whether the code is uniquely decodable or not. A code is the assignment of a unique sequence of characters (a codeword) to each character in an alphabet. A binary code is one in which the codewords contain only zeroes and ones. Here are two possible binary codes for the alphabet {a,c,j,l,p,s,v} in the following table:

```
      Code1 Code2
a     1 010
c     01 01
j     001 001
l     0001 10
p     00001 0
s     000001 1
v     0000001 101
```

The encoding of a string of characters from an alphabet (the cleartext) is the concatenation of the code words corresponding to the characters of the cleartext, in order, from left to right. A code is uniquely decodable if the encoding of every possible cleartext using that code is unique. In the example above, Code 1 is uniquely decodable, but Code 2 is not. For example, the encodings of the cleartexts "pascal" and "java" are both 001010101010. Even shorter encodings that are not uniquely decodable in Code 2 are 01 and 10.

While the agency is very proud of Phil, he unfortunately gives only "yes" or "no" answers. Some members of the agency would prefer more tangible proof, especially in the case of codes that are not uniquely decodable. For this problem you will deal only with codes that are not uniquely decodable.

Task

Given a code, you must determine the encoding having the minimum length (measured in bits) that is ambiguous because it can result from encoding each of two or more different cleartexts.

Input

The input for the code begins with an integer N, $1 <= N <= 30$, on a line by itself, where N is the number of binary codewords in the code. This is followed by N lines each containing one binary codeword string. No codeword will contain more than 30 bits.

Output

Output on a single line the shortest encoding that is not uniquely decodable (with no intervening spaces). If there are several shortest encodings, output any one of them

Sample Input (codes.in):

5
0110
00
111
001100
110

Sample Output (codes.out):

001100110
$1<=N<=5000$
$1<=F\_i<=400$ for all i

Constraints

Time limit: 5 seconds
Memory limit: 16384 Kb

Scoring

There are 10 test cases of 10 points each.
A correct answer scores 10 points.
All incorrect or invalid outputs score 0.

Time limit

1 Second per case.

Water Jugs

Author

Carl Hultquist

Introduction

Farmer John's wife, Martha, needs a certain amount of water in order to make lemonade. However, she needs an exact amount of water otherwise the mixture will not taste very good. She has several water jugs of certain volumes, and can perform the following actions:

- Fill a jug with water at the tap
- Pour water from one jug into another until the destination jug is full, or until the source jug is empty
- Empty the water in a jug down the drain

Task

You need to help Martha obtain the amount of water that she requires in one of the jugs. She also needs to do this using the fewest number of operations possible. If there is more than one way of achieving this, you should pick the method that also uses the least amount of water in total (i.e. least amount of water from the tap).

Example

In this example, Martha has three water jugs, with volumes of 6, 7 and 10 litres of water. She needs to obtain 8 litres of water in one of the jugs. This can be achieved by the following steps:

Fill the 7-litre jug
Pour the 7-litre jug into the 10-litre jug
Fill the 7-litre jug
Pour the 7-litre jug into the 6-litre jug, leaving 1 litre in the 7-litre jug
Pour the 7-litre jug into the 10-litre jug

Input format

The first line of input will contain 2 integers N and L, where N is the number of jugs and L is the number of litres of water that Martha wants. The second line will contain exactly N integers, which indicate the number of litres in each of the N jugs.

Sample input

3 8
6 7 10
Output format
The first line of output must consist of 2 integers O and W, which are the number of operations needed and the total amount of water used. The next O lines must each contain 2 integers S and D, which indicate the source jug and destination jug for the operation. If filling from the tap then S should be -1, and if emptying down the drain then D should be -1. Otherwise, jugs are numbered from 0 to N - 1.

Sample output

5 14
-1 1
1 2
-1 1
1 0
1 2

Constraints

1 <= N <= 6
1 <= K <= 11, where K is the volume of any jug
There is guaranteed to always be a solution for the input data given

Scoring

If the least number of operations is P, and the least amount of water using this number of operations is V, then:

1. If O = P, score S = 20 - 4 * (W - v)
2. If O <> P, score S = 15 - 5 * (O - P)

If S < 0, then your program scores 0.