ACM South American Regional

Collegiate Programming Contest

Warm-up Session

December 13, 1997

Institute of Computing

University of Campinas, Brasil

Problem A

Alternating Sequences

File Names

- **Source File:** altseq.pas or altseq.c or altseq.cpp
- **Input File:** altseq.in
- **Output File:** altseq.out

Statement of the Problem

A sequence of n integer numbers (each number greater than zero) is said to be M-alternating if the sequence is composed by M consecutive segments: the first segment with one element, the second segment with two elements, and so on, the M^{th} segment with M elements. Furthermore, the sequence has to satisfy the following conditions:

- the elements in a given segment have to be all odd or all even;
- if the elements in a segment are all even (odd), then the elements in the following segment have to be all odd (even).

Write a program that, given an integer $n \geq 1$ and a sequence of n positive integers, verifies whether the sequence is M-alternating. The program must output the value for M if the sequence is M-alternating, or output the string "NOT" if the sequence is not M-alternating.

Input Format

The first line informs the value for n; the following one contains n integers corresponding to the sequence. The input file contains several instances which finish with a line containing only a % (percentage symbol).

Output Format

The value of M or "NOT" according to the above conditions. The output file should end outputs for the several input instances by a line with a % (percentage symbol).

Examples

Suppose the following input:

```
10
12 3 7 2 10 4 5 13 5 11
%
8
11 2 3 4 5 77 33 44
%
```

Then the corresponding output is

```
4
%
NOT
%
```

Problem B

Power subsequence

File Names

- **Source File:** power.pas or power.c or power.cpp
- **Input File:** power.in
- **Output File:** power.out

Statement of the Problem

Consider $A = a_1a_2a_3\ldots a_n$ and $B = b_1b_2b_3\ldots b_k$ two finite sequences of characters from the usual 26 letter alphabet of the English language.

B is said to be a subsequence of A if there exists a sequence of integers $i_1, i_2, \ldots, i_k$ such that $1 \leq i_\ell \leq n$ and $a_{i_\ell} = b_\ell$ for all $1 \leq \ell \leq k$. Also, B^m is the sequence formed from B by repeating each of its characters m times consecutively. See the example:

$$B = qwerty, B^3 = qqqwwweeerrrtttyyy$$

Your problem is to write a program that, given two sequences A and B, determines the largest value of $m \geq 0$ such that B^m is a subsequence of A.

Input Format

A file containing the two sequences A and B on a line separated by a space. Each instance of the problem ends with a %.

Output Format

A file containing the value of m.

Examples

Suppose the following input:

```
qwqeqretyqsefgerxrvbr qer
%
ooioioooorapexrrxettioc oioorx
%
```

The corresponding output is:

```
3
%
1
%
```

Problem C

Language Translator

File Names

- **Source File:** verb.pas or verb.c or verb.cpp
- **Input File:** verb.in
- **Output File:** verb.out

Statement of the Problem

Recently, an exploring team discovered a lost tribe in the Amazonian jungle. The natives speak a very simple and regular language, and in order to improve the communication you and your team should write a verb analyzer. The initial contacts with the natives have already allowed the deduction of the following rules:

1. All verbs are regular.
2. All verbs end with a consonant followed by the suffix 'en'.
3. There are only three verb tenses: past, present and future.
4. To conjugate the verb, the natives replace the suffix 'en' by another, according to the following table:

Person	Present	Past	Future
1st	o	ei	ai
2nd	os	es	ais
3rd	a	e	i
4th	om	em	aem
5th	ons	est	aist
6th	am	im	aim

The verb 'campten' has then the following cases:

Person	Present	Past	Future
1st	campto	camptei	camptai
2nd	camptos	camptes	camptais
3rd	campta	campte	campti
4th	camptom	camptem	camptaem
5th	camptons	camptest	camptaist
6th	camptam	camptim	camptaim

The program should analyze a word, verify if it is a verb or not and, if it is a verb, indicate the person and the tense of the original verb.

Input Format

A list of words, one per line.

Output Format

One analyzed word per line, followed by the result of the analysis, as shown in the examples.

Examples

Suppose the following input:

```
casos
porre
corraem
picel
oficina
param
```

Then the Corresponding output is:

```
casos - verb casen, present tense, 2nd person
porre - verb porren, past tense, 3rd person
corraem - verb corren, future tense, 4th person
picel - not a verb case
oficina - not a verb case
param - verb paren, present tense, 6th person
```

Problem D

Chess Player

File Names

- **Source File:** chess.pas or chess.c or chess.cpp
- **Input File:** chess.in
- **Output File:** chess.out

Statement of the Problem

A software company wants to develop a chess program better than Deep Blue. You and your team were hired to write a positional analyzer, and your first task is to generate a list of all valid moves for a given position.

The position is described by a matrix, in accordance with the the chess notation. Rows are numbered from 1 to 8, and columns from 'a' to 'h', as below:

	a	b	c	d	e	f	g	h	
8									8
7									7
6									6
5									5
4									4
3									3
2									2
1									1
	a	b	c	d	e	f	g	h	

White pieces begin the game in rows 1 and 2; black pieces are positioned in rows 7 and 8. The pieces are identified by their first letter. White pieces use uppercase (P for pawn, R for rook, N for knight, B for bishop, Q for queen and K for king), and black pieces use lowercase (p, r, n, b, q and k, respectively).

The special cases of castle and 'en passant' capture will be analyzed later by special routines and don't needed to be considered now. The promotion of pawns, however, should be considered.

Input Format

The position is supplied from row 1 to 8 and, for each row, from column 'a' until 'h'. Empty squares are indicated by the digit 0 (zero). For example, the initial position is:

```
RNBQKBNR
PPPPPPPP
00000000
00000000
00000000
00000000
pppppppp
rnbqkbnr
```

The next line contains W if white should move next, or B otherwise. Each test case ends with a line containing a % as its only character.

Output Format

The kings' moves should be listed first, followed, in order, by the moves from queen, rook, bishop, knight and pawns. Each move should be given in the format <letter><initial column><initial row>-<final column><final row>.

If a piece is captured, the dash ('-') should be replaced by an 'x' followed by the initial of the captured piece (followed by the final column and row). If a pawn is promoted, the movement is followed by the equal sign ('=') and the initial of the new piece.

Each test case should end with a line containing % as its only character.

Examples

Suppose the following input:

```
0000000K
r0000000
0000p000
00000000
00000000
00000000
P000R000
00000000
W
%
```

Then the corresponding output is:

```
Kh1-g1
Re7-b7, Re7-c7, Re7-d7, Re7-f7, Re7-g7, Re7-h7
Re7xpe3, Re7-e4, Re7-e5, Re7-e6, Re7-e8
Pa7-a8=Q, Pa7-a8=R, Pa7-a8=B, Pa7-a8=N
%
```

Chess movement rules

King - can only move one square at a time, in any direction (horizontal, vertical, diagonal). It cannot move to a square where it can be captured. The king can capture any contiguous adversary pieces that aren't protected by another piece. If the king is under attack (check) it should move, or the attacking piece should be captured, or another piece should be placed between the king and the attacking piece. If it cannot be done, the king is under checkmate.

Queen - can move any number of squares, in any direction, but cannot spring over another piece.

Rook - it moves as the queen, but limited to horizontal or vertical movements.

Bishop - it moves as the queen, but limited to diagonals.

Knight - is the only piece that can spring over other pieces. It moves two squares on a row (or column) and then one square on the column (or row). The final square should be empty or occupied by an opponent piece (that is captured).

Pawn - has distinct capture and movement rules. It cannot move backwards, it moves always forward. When in the initial square, it can move one or two squares in a single movement; all other moves are one square at a time. It cannot move if the square in front of it is occupied by another piece. The pawn captures diagonally, also one square at a time. When it reaches the last row (row 8 for white and row 1 for black), it should be immediately replaced by another piece (queen, rook, bishop or knight).