Canadian
Mathematics
Competition

An activity of The Centre for Education
in Mathematics and Computing,
University of Waterloo, Waterloo, Ontario

Canadian Computing Competition

for the  **Awards**

Wednesday, March 5, 1997

Problem A

Sentences

Input file: sent.in Output file: sent.out

Write a program which accepts as input a list of subjects, a list of verbs, and a list of objects, and produces all possible sentences which consist of a subject, a verb, and an object.

Input Specification

The first line of the input file contains a positive integer *n* which is the number of data sets which follow. For each of the *n* data sets, the data begins with three positive integers, one per line, each less than or equal to 20, which represent the number of subjects, verbs, and objects, respectively, which are provided. Following this line are the subjects, one per line in alphabetical order, the verbs, one per line in alphabetical order, and the objects, one per line in alphabetical order. The maximum length of any subject, predicate or object is 25 characters.

Output Specification

The output is to consist of all possible sentences which can be formed using one subject, one verb, and one object, and is to be output in alphabetical order with a period at the end of each sentence. The output for different data sets are to be separated by a single blank line.

Sample Input

```
1
3
3
2
He
The cat
The dog
bit
kicked
saw
him
the mouse
```

Continued on next page

Sample Output

```
He bit him.
He bit the mouse.
He kicked him.
He kicked the mouse.
He saw him.
He saw the mouse.
The cat bit him.
The cat bit the mouse.
The cat kicked him.
The cat kicked the mouse.
The cat saw him.
The cat saw the mouse.
The dog bit him.
The dog bit the mouse.
The dog kicked him.
The dog kicked the mouse.
The dog saw him.
The dog saw the mouse.
```

Problem B

Nasty Numbers

Input file: nasty.in Output file: nasty.out

We will call a positive integer "Nasty" if it has at least two pairs of positive integer factors such that the difference of one pair equals the sum of the other pair.

For example, 6 is nasty since $6 \times 1 = 6$, $2 \times 3 = 6$, and $6 - 1 = 2 + 3$;
and 24 is also nasty since $12 - 2 = 6 + 4$.

Write a program which accepts as input a list of positive integers and determine if each one is nasty or not.

Input Specification

The input file is a list of positive integers, one per line. The first number in the list is the number of integers to be tested, and is at most 20. The integers to be tested are all less than 32001.

Output Specification

The output file should contain one line for each test value. Each line is to contain the test value and whether it is nasty or not.

Sample Input

4
6
24
30420
10078

Sample Output

6 is nasty
24 is nasty
30420 is nasty
10078 is not nasty

Problem C

Double Knockout Competition

Input file: dkc.in Output file: dkc.out

In a number of sports, a championship may be determined by a double knockout competition. A team is eliminated on its second loss, so the winner is the last remaining team with one or fewer losses. The competition is played is a series of rounds: in each round, teams that have not been eliminated are paired subject to the constraint that an undefeated team never plays a team with one loss. As many teams as possible are paired in each round. After a number of rounds only two teams remain. These teams play in a round by themselves, although one is undefeated and the other is not. If neither is eliminated, they play again in a final round. For our analysis we assume that this extra round is always necessary.

a) Write a program to report on a Double Knockout Competition.

Input Specification

The first line of input contains one positive integer n which is the number of test cases which follow it. The next n lines each contain one positive integer t, $t < 32768$, which is the number of teams in the competition for that test case.

Output Specification

For each case there should be an initial line which has the form:

Round 0: 2 undefeated, 0 one-loss, 0 eliminated

This is followed by a similar line for each round of the competition, followed by a single line saying how my rounds were played. The output for different test cases is to be separated by a single blank line.

Sample Input

1
2

Continued on next page

Sample Output

Round 0: 2 undefeated, 0 one-loss, 0 eliminated
Round 1: 1 undefeated, 1 one-loss, 0 eliminated
Round 2: 0 undefeated, 2 one-loss, 0 eliminated
Round 3: 0 undefeated, 1 one-loss, 1 eliminated
There are 3 rounds.

b) If there are $t = 2^{(2^k)}$ teams, where k is an integer, how many rounds are played in the tournament?

c) How many games are played in an t team competition?

Problem D

Dynamic Dictionary Coding

Input file: ddc.in Output file: ddc.out

A common method of data compression, "dictionary coding", is to replace words in a text by numbers indicating their positions in a dictionary. Static dictionary coding, in which the dictionary is known in advance, can be problematic, as it is necessary to have the dictionary available to understand the coded text. Dynamic dictionary coding avoids this problem by deriving the dictionary from the text to be compressed. The text is processed from beginning to end, starting with an empty dictionary. Whenever a word is encountered that is in the dictionary, it is replaced by a number indicating its position in the dictionary. Whenever a word is encountered that is not in the dictionary, it appears as-is in the compressed text and is added to the end of the dictionary.

(a) You are to implement dynamic dictionary coding.

Input Specification

The first line of the input file contains a positive integer which is the number of sets of text which are to be compressed. Each set of text will consist of several lines containing text made of lower case letters and spaces only. You may assume that no word is longer than 20 letters, that no input line is longer than 80 characters, and that there are no more than 100 input lines. The sets of text are separated by a single blank line, and are to be compressed individually.

Output Specification

The output file should contain the sets of text input compressed using dynamic dictionary coding as described above. Lineation and spacing should be preserved exactly as in the input file with the different sets of compressed text separated by a single blank line.

Sample Input

```
1
the cat chased the rat while
the dog chased the cat into the rat house
```

Sample Output

```
the cat chased 1 rat while
1 dog 3 1 2  into 1 4 house
```

Continued on next page

(b) When a word is first encountered, why is it added to the dictionary but not encoded using its newly created position in the dictionary?

(c) Suppose the input file contained numbers as well as words. Explain the difficulty that would arise and suggest an alteration to the method that would overcome the difficulty.

Problem E

Long Division

Input file: div.in Output file: div.out

In days of yore (circa 1965), mechanical calculators performed division by shifting and repeated subtraction. For example, to divide 987654321 by 3456789, the numbers would first be aligned by their leftmost digit (see (1) below), and the divisor subtracted from the dividend as many times as possible without yielding a negative number. The number of successful subtractions (in this example, 2) is the first digit of the quotient. The divisor, shifted to the right (see (2) below), is subtracted from the remainder several times to yield the next digit, and so on until the remainder is smaller than the divisor.

```
    987654321
–   3456789          ← First successful subtraction (1)
    ========
    641975421
–   3456789          ← Second successful subtraction
    ========
    296296521        ← remainder
–   3456789          ← unsuccessful subtraction
    ========
    negative

    296296521
      3456789        (2)
    ========
    261728631
          etc.
```

(a) Write a program to implement this method of division.

Input Specification

The first line of the input file contains a positive integer n, $n \le 20$, which represents the number of test cases which follow. Each test case is provided on a pair of lines, with the number on the first line being the dividend, and the number on the second line being the divisor. Each line will contain a positive integer of up to 80 digits in length.

Continued on next page

Output Specification

For each pair of input lines, your output file should contain a pair of lines representing the quotient followed by the remainder. Output for different test cases should be separated by a single blank line. Your output should omit unnecessary leading zeros.

Sample Input

```
987654321
3456789
33
11
11
33
```

Sample Output

```
285
2469456

3
0

0
11
```

(b) If the dividend is *n* digits long and the divisor is *m* digits long, derive a formula in terms of *n* and *m* that approximates the maximum number of single-digit subtractions performed by your program.