

Problem S: The Dole Queue

In a serious attempt to downsize (reduce) the dole queue, The New National Green Labour Rhinoceros Party has decided on the following strategy. Every day all dole applicants will be placed in a large circle, facing inwards. Someone is arbitrarily chosen as number 1, and the rest are numbered counter-clockwise up to N (who will be standing on 1's left). Starting from 1 and moving counter-clockwise, one labour official counts off k applicants, while another official starts from N and moves clockwise, counting m applicants. The two who are chosen are then sent off for retraining; if both officials pick the same person she (he) is sent off to become a politician. Each official then starts counting again at the next available person and the process continues until no-one is left. Note that the two victims (sorry, trainees) leave the ring simultaneously, so it is possible for one official to count a person already selected by the other official.

Write a program that will successively read in (in that order) the three numbers (N , k and m ; $k, m > 0, 0 < N < 20$) and determine the order in which the applicants are sent off for retraining. Each set of three numbers will be on a separate line and the end of data will be signalled by three zeroes (0 0 0).

For each triplet, output a single line of numbers specifying the order in which people are chosen. Each number should be in a field of 3 characters. For pairs of numbers list the person chosen by the counter-clockwise official first. Separate successive pairs (or singletons) by commas (but there should not be a trailing comma). Example:

Sample input

```
10 4 3
0 0 0
```

Sample output

```
△△4△△8,△△9△△5,△△3△△1,△△2△△6,△10,△△7
```

where \triangle represents a space.

Problem T: Perfect Numbers

For any number N , if the sum of its divisors (other than itself) equals N the number is called perfect; if the sum is less than N it is called deficient; if the sum is greater than N it is called abundant. Thus:

- 6 has divisors 1,2,3; sum = 6; therefore 6 is perfect.
- 8 has divisors 1,2,4; sum = 7; therefore 8 is deficient.
- 24 has divisors 1,2,3,4,6,8,12; sum = 36; therefore 24 is abundant.

Write a program that will read in a number and classify it as perfect, deficient or abundant.

Input will consist of a series of numbers, one per line. The input will be terminated by zero (0).

Output will consist of one line for each number in the input file and will echo the original number followed by the sum (both right justified in fields of width 4 columns) followed by one of the words 'perfect', 'deficient' or 'abundant' as shown in the example.

Sample input

```
6
8
24
0
```

Sample output

```
  4   8           20
6    6    perfect
8    7  deficient
24   36  abundant
```

Problem U: Telephone Tangles

A large company wishes to monitor the cost of phone calls made by its personnel. To achieve this the PABX logs, for each call, the number called (a string of up to 15 digits) and the duration in minutes. Write a program to process this data and produce a report specifying each call and its cost, based on standard Telecom charges.

International (IDD) numbers start with two zeroes (00) followed by a country code (1–3 digits) followed by a subscriber’s number (4–10 digits). National (STD) calls start with one zero (0) followed by an area code (1–5 digits) followed by the subscriber’s number (4–7 digits). The price of a call is determined by its destination and its duration. Local calls start with any digit other than 0 and are free.

Input will be in two parts. The first part will be a table of IDD and STD codes, localities and prices as follows:

Code Δ Locality name\$price in cents per minute

where Δ represents a space. Locality names are 25 characters or less. This section is terminated by a line containing 6 zeroes (000000).

The second part contains the log and will consist of a series of lines, one for each call, containing the number dialled and the duration. The file will be terminated a line containing a single #. The numbers will not necessarily be tabulated, although there will be at least one space between them. Telephone numbers will not be ambiguous.

Output will consist of the called number, the country or area called, the subscriber’s number, the duration, the cost per minute and the total cost of the call, as shown below. Local calls are costed at zero. If the number has an invalid code, list the area as “Unknown” and the cost as –1.00.

Sample input

```
088925 Broadwood$81
03 Arrowtown$38
0061 Australia$140
000000
031526      22
0061853279  3
0889256287213  122
779760 1
002832769 5
#
```

Sample output

<u>1</u>	<u>17</u>	<u>51</u>	<u>56</u>	<u>62</u>	<u>69</u>
031526	Arrowtown	1526	22	0.38	8.36
0061853279	Australia	853279	3	1.40	4.20
0889256287213	Broadwood	6287213	122	0.81	98.82
779760	Local	779670	1	0.00	0.00
002832769	Unknown		5		-1.00

Problem V: Maximal Subsequences

Given a sequence of N integers ($1 \leq N \leq 100$), a maximal subsequence (for the purposes of this problem) is the subsequence with the highest sum. Thus for the sequence:

1, 2, 8, -7, 3, 5, -20, 2, 1

the maximal subsequence runs from 1 to 6 with a sum of 12 since no other subsequence has a higher sum.

Write a program that will read in a sequence and find the maximal subsequence. Your program should be able to produce an answer in a reasonable time even for a sequence of 100 elements.

Input will consist of number (N) on a line by itself, followed by N numbers on as many lines as necessary. There will be a number of such sets of lines terminated with $N = 0$. All sequences in the input will have a positive maximal subsequence.

Output will consist of the start and end indices and the resulting sum for each given sequence all right justified in fields of 10 characters, one result per line. If there is more than one maximal subsequence then the shortest one (the one containing the fewest numbers) should be reported. If there is more than one maximal subsequence of shortest length then the one with the lowest start index should be reported.

Sample input

```
11
1 2 8 -7 3 5 -20 2 1 -2 -3
4
-10 105 6 -2
0
```

Sample output

```
1         6         12
2         3         111
```

Problem W: Ugly Numbers

Ugly numbers are numbers whose only prime factors are 2, 3 or 5. The sequence

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...

shows the first 11 ugly numbers. By convention, 1 is included.

Write a program to find and print the 1000'th element in this sequence. There is no input to this program. Output should consist of a single line as shown below, with <number> replaced by the number computed.

Sample output

The 1000'th ugly number is <number>.

Problem X: Story Sum

Consider the following sentence:

‘There are: a 1’s, b 2’s, c 3’s, d 4’s, e 5’s, f 6’s, g 7’s, h 8’s, i 9’s and j 0’s in this sentence.’

Write a program that will find the values of a through j that makes the above sentence correct. Each of the numbers a through j must be in the range 1 to 9 inclusive (there is a second solution where one of the values is outside this range, but that is **not** the one you are required to find).

There is no input to this program.

Output will consist of a single line containing the required 10 numbers each right justified in a field of 3 characters.

Problem Y: The Expressive Grocer

The local grocer, being of a somewhat inventive spirit, is trying a new way of marking prices on all her stock. Rather than simply marking each item with a price, like \$3.50, she labels items with their prices relative to other items in the shop. For example, butter may be labelled “margarine+10c”, margarine may be labelled “coffee - \$1.25”, etc. A few items are marked with actual prices.

You have heard that her prices are very reasonable, and wish to prepare a list of what everything costs in a more conventional fashion, such as “butter \$2.25”. You are to write a program to perform this task.

Input will contain a series of lines like the following:

```

milk = sugar - 125
glucose = 225
sugar = glucose + 10
coffee = tea
#

```

where all numbers are in cents. The end of the input file will be marked by a line containing a single ‘#’. When all prices have been read, print a list, in alphabetical order, of all items mentioned in the input together with their prices as shown below. If an item’s price cannot be deduced from the input data, give it as “\$blank”.

The output corresponding to the above input would be:

```

1
13
coffee  $blank
glucose  $ 2.25
milk     $ 1.10
sugar    $ 2.35
tea      $blank

```

You may make the following assumptions about the input data.

1. All items have names made up of at most 10 lower-case alphabetic characters only.
2. There are fewer than 100 items involved.
3. Input lines either give the price of an item directly (in cents), or give it as equal to some other item, or give it as equal to some other item plus or minus a number of cents.
4. Spaces may appear anywhere on the input lines except within item names or numbers.
5. Numeric values are always given in cents, and are always less than 1000.
6. All input is syntactically correct.
7. No item appears more than once to the left of an equals sign.

Problem Z: Alphabet Subsets

Words are anagrams of each other if they are the same length and contain the same letters, thus 'pots', 'stop', 'spot' and 'opts' are anagrams. Words can be said to share the same base alphabet if they have all their letters in common, even if some of them are repeated, thus 'curse' and 'rescue' share the same base alphabet, but 'cure' does not share a base alphabet with either because it lacks an 's'.

Write a program that will repeatedly read pairs of words and determine whether or not they share a base alphabet or not.

Input will consist of a series of lines each containing a single word of less than 20 lower case characters. Successive lines will form the relevant pairs. The end of input will be signalled by a line containing a single '#'.

Output will consist of a single line for each pair of words. The first word of the pair will start in column 1, the second word will start in column 21 and starting in column 41 the word 'yes' or 'no' depending on whether or not the words share a base alphabet.

Sample input

```
curse
rescue
curse
cure
#
```

Sample output

```
curse           rescue           yes
curse           cure             no
```