2006 Canadian Computing Competition Day 1, Question 1

Input file: `cntower.in`
Output file: `cntower.out`
Source file: `n:\cntower\cntower.{c, cpp, pas}`

CN Tower

Christy C. Coder is traveling to Waterloo for a programming competition. On the way, she stops in Toronto to do some sightseeing. The unfortunate thing about travelling is that everyone back home expects her to bring back pictures of everything. Christy hates taking pictures: it makes her look like such a tourist! Fortunately, Christy has a plan to make her picture-taking quite painless.

At 553 m tall, CN Tower is the world's tallest free-standing building. 351 m up the tower is the "360" rotating restaurant, which rotates a full 360 degrees every 72 minutes. From there, Christy can see the whole city, and take close-up pictures of all the landmarks using her fancy new 100x optical zoom camera. Since the restaurant itself rotates, she only needs to stand in one place to take pictures in all directions.

The waiters insist that she order something or leave, and Christy is not interested in any of the items on the menu. Therefore, she must act quickly before she gets kicked out. Given the locations of the landmarks of which Christy wants to take a picture, your task is to determine the minimum amount of time that she must spend in the restaurant in order for it to rotate enough to bring all the landmarks in view. Assume that Christy always points her camera exactly perpendicular to the window to minimize distortion due to the glass. Note that multiple landmarks may occupy the same (angular) position, and these landmarks would only require one photograph to capture them.

Since the restaurant staff only realize she is a tourist once she starts taking pictures, we begin measuring the time required once she takes her first picture. Therefore, Christy can move to any position in the restaurant without hassle from the restaurant staff and begin taking pictures from there.

Input

The first line of input consists of an integer n ($2 \leq n \leq 1000$), the number of landmarks Christy wants to photograph. Each of the following n lines specify a landmark. Each landmark specification consists of the landmark name (a string of uppercase and lowercase letters of length at most 40 characters), a space, and the compass angle d, in degrees, to the landmark from the CN Tower (0 = north, 90 = east, 180 = south, 270 = west). Note that d is a real number which satisfies $0 \leq d < 360$, with d specified to the hundredth of a degree (i.e., at most two decimal places).

Output

A single integer, the minimum number of seconds that Christy must remain in the restaurant. If the time is not an integer number of seconds, round it up to the nearest second (i.e., take the ceiling of the number).

Sample Input

```
5
CasaLoma 231
OntarioParliament 123
SkyDome 75
RoyalYorkHotel 340
PearsonAirport 165
```

Output for Sample Input

```
3012
```

2006 Canadian Computing Competition
Day 1, Question 2

Input file: `rj.in`
Output file: `rj.out`
Source file: `n:\rj\rj.{c, cpp, pas}`

R & J

Years ago, Romy and Jules were separated by their parents and forbidden to see each other ever again. However, using a tin-can telephone, their love survived and they were able to maintain their relationship. But, as time passed, some things changed and some things remained the same. Romy's great (times 6) grand-child, Julia (from the planet Mars), is in love with Romian (from the planet Epsilon 186-Beta), but because of their parents hatred is not able to speak with him. To prevent Julia and Romian from seeing each other, their parents put them both on spaceships with no long-range communication facilities. Thus, Julia and Romian must use a laser to send each other messages using morse-code. In Romian and Julia's time, messages can be sent infinite distances through space using a laser, as long as the route between the sender and receiver is not blocked. Since this is the future, lasers travel instantly across the universe. Therefore, you may assume that planets and spaceships are stationary.

For this question, you are to determine whether Romian and Julia can communicate with each other (by laser) when given the 3-dimensional grid coordinates of their spaceships. The problem is that their communication laser is often blocked by intervening planets. Thus, they are truly star-crossed lovers.

Input

For input you will be given, on two lines, the grid coordinates of Romian's and Julia's spaceships. Each coordinate will consist of three, whitespace separated, integer values. Then, following the locations of their spaceships, the input will consist of a line containing a single integer, n ($1 \leq n \leq 2000$), indicating the number of planets in their current sector of space. The coordinates of the n planets follow on the next n lines. Each planet is represented by 4 integer values, separated by whitespace. The first three are the coordinates of the planet's center and the fourth is the planet's radius. It may be helpful to know that a planet with centre (a, b, c) and radius r can be specified by the equation $(x-a)^2+(y-b)^2+(z-c)^2 = r^2$. You may assume that $0 \leq r \leq 5000$ and that for any coordinate (a, b, c), $0 \leq |a|, |b|, |c| \leq 5000$.

You may also make the following assumptions:

- Neither Romian nor Julia are within 10^{-8} units of a planet (since they would crash into it), nor are they within a planet (since that is impossible).
- Romian and Julia do not occupy the same position in space (otherwise, their spaceships

have crashed into each other and they will be consumed by an intergalactic explosion of epic proportions).

- If the laser hits a planet, it would also have hit a planet with a radius which is 10^{-8} units smaller. If the laser misses a planet, it would have avoided a planet which was 10^{-8} units larger.

Output

As output, you are to print a single integer value, b, where $0 \leq b \leq n$. This value indicates the number of planets that block the laser. Output a zero if no planets block the laser.

Sample Input

```
100 100 100
-100 -100 -100
2
0 0 0 2
50 60 -50 5
```

Output for Sample Input

```
1
```

2006 Canadian Computing Competition
Day 1, Question 3

Input file: `codec.in` (for `compress`), `compress.out` (for `decompress`)
Output file: `compress.out` (for `compress`), `codec.out` (for `decompress`)
Source files: `n:\codec\compress.{c, cpp, pas}`
`n:\codec\decompress.{c, cpp, pas}`

Codec

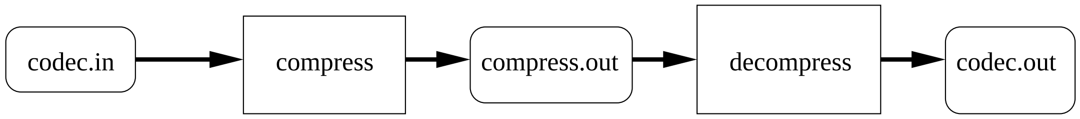
The problem of lossless data compression is to transform some data into a compressed form such that:

(a) the original can be reproduced exactly from the compressed form.

(b) the compressed form is as small can reasonably be achieved.

You are to write two programs – `compress` that performs lossless compression and `decompress` that reproduces the original data from the compressed form. The data to be compressed will be plain English text represented using printable ASCII characters (i.e., all characters with ASCII values between 32 and 126 inclusive). The compressed form is a string of binary bits. For convenience, we will represent this string of bits as a character string containing only 0s and 1s.

`compress` reads the original data from the file `codec.in` and writes the compressed form `compress.out`. `decompress` reads the compressed data from a file called `compress.out` and writes the corresponding original data to `codec.out`. Of course, `codec.in` and `codec.out` must be the same file. Pictorially, we have the following flow of information:



`compress` must output only 0s and 1s and `decompress` must exactly reverse the effects of `compress`. That is, condition (a) above must hold for any English text. If `compress` and `decompress` meet these criteria, your score will be determined by the relative size of the input and output by

$$\text{score (as \%)} = 50 \cdot \sqrt{\frac{8c - b}{c}},$$

where c is the total number of characters in the original text and b is the number of bits in the compressed form. Note that scores may exceed 100%, but scores that are less than 0 will be given 0% (i.e., no negative marks will be given, but bonus marks may be awarded).

Discussion and Hints

It is well known that any ASCII character can be represented using 8 bits. Such a representation would achieve a score of 0 using the formula above. Since there are fewer than 128 possible symbols in the input, it is possible to represent each one with 7 bits. Such a representation would receive a score of at least 50%.

A smaller representation can be achieved, with high probability, by observing that some letters are more common than others. Suppose we estimate that a character α occurs with probability p_α in a given context. The best possible code will use $-\log_2(p_\alpha)$ bits to represent that character. If one estimates p_α one can construct a *prefix* code with about $-\log_2(p_\alpha)$ bits for each character in the following manner:

- build a binary tree with one leaf for each character α
- organize the tree so that the depth of α is approximately $-\log_2(p_\alpha)$
- use a binary representation of the path (0=left, 1=right) to represent α in the compressed data.

One way to estimate p_α is simply to compute the fraction of characters equal to α in a sample of data similar to that to be compressed. Another is to use an adaptive method, in which the data is compressed one character at a time, and the sample consists of the text already compressed. A sample of English text is available in the file `sampleText.txt`.

It is also possible to estimate p_α using the context in which it occurs; for example, in English a "q" is very likely to be followed by a "u" (e.g., quick, quack, quit, quiz, but not qiviut, which happens to be the wool of a musk-ox).

Use this information, or any other information at your disposal, to build the best Compressor and Decompressor you are able.

Input

The input to `compress` will consist of n characters ($1 \le n \le 1000000$), as described above.

The input to `decompress` will consist of m 0s and 1s ($1 \le m \le 8n$).

Output

The output of `compress` will be a sequence of 0s and 1s, with no other characters (i.e., no newline characters should be outputted).

The output of `decompress` is a sequence of at most 1000000 uppercase letters, lowercase letters, spaces, newlines and punctuation symbols.

Sample Input (to `compress`)

```
To be or not to be?
```

Possible Output for Sample Input (`compress`)

```
0111001000010110000100110100001100010011110000011110010000101101111 1
```

Sample Input (to `decompress`)

```
0111001000010110000100110100001100010011110000011110010000101101111
```

Output for Sample Input (`decompress`)

```
To be or not to be?
```

Explanation

The sample compressor systematically uses the following codes for each of the input characters:

<space>	000
o	010
n	01100
r	01101
T	01110
t	011110
?	011111
b	10
e	11

The compressed output uses these codes, as shown below:

```
0111001000010110000100110100001100010011110000011110010000101101111
TTTTTooo   bbee   ooorrrrr   nnnnnooottttt   tttttooo   bbee??????
```