Day 1 Question 1

Palindromes

Input file: proba.in

Output file: proba.out

A palindrome is a sequence of characters that reads the same forwards and backwards. e.g. RADAR, MADAMIMADAM
You are to write a program which finds the longest palindrome in a given string of characters.

Input

The input file will begin with a line containing one positive integer *n*, the number of strings to be tested, followed by *n* lines each containing one string of characters of up to 25,000 characters in length and terminated with a blank. The input strings will contain upper case letters only.

Output

Your program should output a pair of lines for each test case with the palindrome on the first line and the length of the palindrome on the second. In the event of a tie for longest, any of the palindromes in the tie may be reported.

Sample Input

```
1
AHAHJHFYUBNMLOIUYTRERTYUIOLMNBAGWOIS
```

Output for sample input

```
BNMLOIUYTRERTYUIOLMNB
21
```

Day 1 Question 2

Alien Invasion

Input file: probb.in

Output file: probb.out

Earth is being invaded by space aliens. Earth defence forces have rallied a number of anti-spacecraft guns. However, they have a bug in their aiming hardware: initially they are aimed straight up, and this aim can only be adjusted downward.

Thousands of alien craft are streaking towards Earth as we speak -- and yes, some of them are even aimed at Canada. The Earth defence forces must now come into play. Each gun can fire as many shots as necessary, and can be re-fired as often and as quickly as necessary, but only to a lower setting. Thus if a spacecraft came in at height 3 and then another at height 2, one gun could eliminate both, but could not if they came in the other order. The Earth has only a finite number of guns and it is unknown how many alien craft are coming in. Thus they need a way to minimize the number of guns for a given set of incoming alien craft. Guess what? This is where you come in!

Input

The data will consist of several sets of data. The first line of each set will contain one positive integer *n* (*n* < 100000), where *n* is the number of incoming alien craft. The next *n* lines will contain one floating point number giving the heights of the incoming alien craft in order of arrival (ie, the order the guns must eliminate them). The last line in the data file will contain only zero, ie *n* = 0.

Output

For each set other than the final, *n* = 0, case, one integer specifying the minimum number of guns required to eliminate EVERY alien craft is to be output.

Sample Input

```
10
4.0
2.0
3.0
4.0
5.0
3.0
1.0
4.0
2.0
5.0
0
```

Output for sample input

```
4
```

Day 1 Question 3

Porous Stone

Input file: probc.in

Output file: probc.out

The following can be viewed as modeling the diffusion of a liquid (e.g. some toxic substance) through a porous medium (e.g. the ground) and asking whether it will reach some region (e.g. the water supply). However, we will simplify the presentation and pose the problem in just two dimensions.

We are given a (2k+1) by (2k+1) grid, say (G[i,j]: i=-k...k, j=-k...k). You are to write a program to determine by simulation, the probability that the boundary can be reached from the starting point (0,0). This is done by doing *t* trials, counting the number that succeed in reaching the boundary, and then dividing this by *t* to find the probabilty of escape. A boundary point is any point G[i,j] such that either i or j is k or -k.

On any one trial, the probability that there is a gap between a grid point (i,j) and its neighbour (i+1,j) is determined by *p* and this probability stays constant for one trial. This is also true for the other three neighbours of point (i,j). The value for *p* is to be input but the probability of a gap will be generated by a random number generator.

Note: Efficiency is of some concern here and re-initializing the grid for each trial may be too costly. Programs will only be allowed to run for 2 minutes.

Input

The input file will begin with a line containing a single positive integer, *n*, the number of cases to be run. This will be followed by *n* sets of 3 lines, each containing one value:

first -
  k, a positive integer giving the grid size as (2k+1) by (2k+1)
second -
  p, a real value giving the probability of being able to make a given move,
third -
  t, a positive integer giving the number of trials to be run.

Output

Your output should consist of *n* lines of the form

  k p t esc

where *k*, *p*, and *t* are as defined above, and *esc* is the proportion of trials that led to an escape, that is the probability of escape. You should give the value of *esc* to two decimal places.

Sample input

```
1
10
0.45
100
```

Possible Output for Sample Input

```
10  0.45  100  0.47
```

Note: If you run your program several times with the data given above, you would not expect to get exactly the same value of *esc* for each run.