ACM-UCLA Programación Creativa 2011
Practice One Programming Contest

March 19, 2011

Universidad Centroccidental Lisandro Alvarado

Contest session

This problem set contains 6 problems; pages are numbered from 1 to 7.

Problem A
Aftermath

Source file name: `aftermath.c`, `aftermath.cpp` *or* `aftermath.java`
You must read from standard input and write to standard output.

Here comes the day, from far distances people has just came closer to what would become the ultimate match, in one of the greatest tournaments of our current history. There're N competitors and there is a match for each pair of competitors. You were wondering about how many matches are going to occur.

Input

The input is described by several lines, each one with one integer N, $1 \leq N \leq 10^6$. The input ends $N = 0$, this case shouldn't be processed.

Output

For each test case you must print a line with the number of matches that are going to happen in the tournament.

Sample input

```
3
8
0

```

Sample output

```
3
28

```

Problem B

Back to the Computation School

Source file name: `back.c`, `back.cpp` *or* `back.java`
You must read from standard input and write to standard output.

We have stepped back in time and went back to the beginning of computer science. This is an easy problem if you can remember bitwise operations and use a bit of logic, Russian logic (evil laugh).

Think of an 8-bit integer K, which you can represent in hexadecimal as K = 0xAB, then ~K = 0xBA, you should know that A and B can be any digit in hexadecimal representation and the '~' operation is defined as:

```
~1 = 0
~0 = 1
```

Input

This problem has no input.

Output

You must generate a list of all the 8-bit integers that satisfy the condition described before.

Sample output

```
15

.

.

.

240

```

Problem C

Compression Algorithm

Source file name: `comp.c`, `comp.cpp` *or* `comp.java`
You must read from standard input and write to standard output.

You were hired to decompress strings to which we applied a compression algorithm. This compression algorithm is very easy and that is called LLE (Leo Length Encoding). And easy form to view this data compression is consecutive occurrences of the same character are replaced by a single character followed by its frequency. As an example, the string 'AABBBBDAA' would be encoded to 'A2B4D1A2', quotes for clarity.

Input

The input consists of several cases, each in a line and the string do not exceeds of 500 characters. The last case is followed by the line '#' (without the quotes.)

Output

You must print the decoded string for each case. You can assume that the outputted string won't be longer than 1000 characters.

Sample input

```
A2B4D1A2
A12
A1B1C1D1
#

```

Sample output

```
AABBBBDAA
AAAAAAAAAAAA
ABCD

```

Problem D

Dynamic Probability

Source file name: `dp.c`, `dp.cpp` *or* `dp.java`
You must read from standard input and write to standard output.

Let's suppose this scenario. We have n balls, and a cranky machine that may, or may not take each of the balls. The machine, let's call it the *Brain*, is a supercomputer in current world. For each ball, the *Brain* could decides for take it or not, but we're not sure about why it would do so, however we're aware there's a probability for which the ball could be took. Now, given k, what is the probability of being selected k balls from all of them?

Input

The input includes several cases. The first line for each case includes two integers, n and k, $1 \le n \le 1000$, $0 \le k \le n$. The second line has n floating-point values, where the i-th value is the probability of the i-th ball of being selected. The last case is followed by two zeroes.

Output

For each case print the probability of being selected exactly k balls by the *Brain*, it must be shown with 3 digits of precision.

Sample input

```
3 2
0.4 0.4 0.4
0 0

```

Sample output

```
0.288

```

Problem E

Encryption

Source file name: `encryption.c`, `encryption.cpp` *or* `encryption.java`
You must read from standard input and write to standard output.

Once upon a time, during the Greek era, one men of the Hellenistic Period that existed and had wonderful powers to encrypt documents during wars like of the Romans against Perseus of Macedonia or the Carthaginians in the Third Punic War.

This man was responsible for a useful tool in telegraphy which allowed letters to be easily signaled using a numerical system. This idea also lends itself to cryptographic manipulation and steganography.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

This was known as the "Megalopolis-Encryption", where the letters of the alphabet were arranged left to right, top to bottom in a 5 x 5 square, (when used with the modern 26 letter alphabet, the letters "I" and "J" are combined). Five numbers were then aligned on the outside top of the square, and five numbers on the left side of the square vertically. Usually these numbers were arranged 1 through 5. By cross-referencing the two numbers along the grid of the square, a letter could be deduced. This is a complex idea because when you need encrypt the word *Megalopolis* the result is *32 15 22 11 31 34 35 34 31 24 43*.

Input

The first line of the standard input contains one integer T ($T < 100$) which is the number of test cases. In each of the next T lines there is a string n that you must encipher (it could contains spaces, which you don't have to encipher; apart from spaces it will be only made of uppercase letters - 'A' to 'Z').

Output

For each test print the ciphered string.

Sample input

```
3
I LIKE THE PROGRAMMING CONTEST
ACM RULES
I DONT LIKE THE SYSTEMS SUBJECT

```

Sample output

```
24 31 24 25 15 44 23 15 35 42 34 22 42 11 32 32 24 33 22 13 34 33 44 15 43 44
11 13 32 42 45 31 15 43
24 14 34 33 44 31 24 25 15 44 23 15 43 54 43 44 15 32 43 43 45 12 24 15 13 44

```

Problem F

Fire Brigade

Source file name: `fire.c`, `fire.cpp` *or* `fire.java`
You must read from standard input and write to standard output.

Do you remember our old friend Roy? Well, he again needs a little help from you, but since nobody could help him last time, don't worry, he knows this task has to be a lot easier.

Well, as you now, Roy has "issues" with his mind; lately he got a recurring dream. In his dream he belongs to the fire brigade, but that's irrelevant to the story. The sequence begins with a couple of fire spheres, they just keep growing and growing, destroying anything in their way, until they touch each other and collapse, which become all the world to just nothing, and only few minutes later he appears to be just looking to a beautiful pencil-made picture aside the bed, with trees around the grass looking at the sun shining. Since the nightmare just comes in again and again, he was wondering about how much time he has to wait to its end, so he asks you the distance between the two spheres.

Input

The input has several test cases. Each test case consists of a single line with only eight integers $X_1, Y_1, Z_1, R_1, X_2, Y_2, Z_2, R_2$ $(-1000 < X_1, Y_1, Z_1, X_2, Y_2, Z_2 < 1000; 0 < R_1, R_2 < 1000)$ where X_1, Y_1, Z_1 represent the position coordinates of the center of first sphere, and R_1 its radius. Likewise X_2, Y_2, Z_2 and R_2 indicates the center and radius of the second sphere. You can assume they don't intersect. The input ends with an EOF marker.

Output

For each test case print one line containing a floating-point value with the distance between the two spheres, with 2 digits of precision.

Sample input

```
0 0 0 5 10 10 10 2
3 1 2 5 8 14 8 1

```

Sample output

```
10.32
9.17

```