

**2011-0912 UM Test Online Programming Contest  
March 12, 2010**

**Sponsored by:  
University of Michigan**

Rules:

1. There are a few questions to be tried out any time during the practice contest.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. The allowed programming languages are C, C++ and Java.
4. All programs will be re-compiled prior to testing with the judges' data.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
6. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
7. All communication with the judges will be handled by the PC2 environment.
8. Judges' decisions are to be considered final. No cheating will be tolerated.

## **A. Hello, World!**

In the world where you only need 640Kb of RAM, there exists a certain compiler that takes ASCII code as input and produces machine code as output. To do so it uses a two-phase compiler. The first phase builds a symbol table. Second phase uses that symbol table to create meaningful machine code.

Your first task is to realize that this entire problem statement has little to do with the actual task involved! Your next task is to test out this compiler by writing a C, C++ or Java code to output a string of text exactly as described below.

### **Input**

There is no input for this problem.

### **Output**

For output you are to print the following text on a line by itself: "Hello, World!", followed by the end of line symbol (new line character). No quote marks are allowed in the output.

### **Sample Output**

```
Hello, World!
```

## **B. This Way to Poland!**

May 14 through 18<sup>th</sup> is the World Finals in Warsaw, Poland. To get there, you need to navigate yourself through various routes, where each route has a different length. You have the map. Your team wants to know exact total distance to get there. What do you do?

### **Input**

The first integer number  $n$  indicates the number of routes to follow. The next  $n$  integer numbers are route length. There are multiple test cases. The last case will have number 0 printed by itself on a line

### **Output**

For each input, print out the sum of the  $n$  numbers followed by the end of line symbol

### **Sample Input**

```
3 1 2 3
4 5 2 4 2
0
```

### **Sample Output**

```
6
13
```

## C. Numbers

This problem is to test your I/O contest-style.

You are given a decimal number. Output that number formatting it to having 4 digits after the dot, including zeroes.

### Input

Input will consist of multiple problem instances. The first line will consist of a single positive or negative decimal number. The last line will be 0 and is not to be processed.

### Output

For each problem instance, you are to produce one line of output in the format of a decimal number with 4 digits after the dot, including zeroes.

### Sample Input

```
92.353235
32094.5323323509
5
5.3
0
```

### Sample Output

```
92.3532
32094.5323
5.0000
5.3000
```

## D. Sorting Words

This problem is to test your I/O contest-style.

You are given some words. Sort them and output them according to the instructions below.

### Input

Input will consist of multiple problem instances. The first line will consist of a single positive integer  $n \leq 20$ , which is the number of problem instances. The input for each problem instance will be on two lines. The first line will consist of a positive integer  $m \leq 10$  and the second line will consist of  $m$  lower-case words, separated by a single space and each containing no more than 30 characters.

### Output

For each problem instance, you are to produce one line of output in the format:

Case i: <sorted list of words>

The value of  $i$  is the number of the problem instance (we start numbering at 1) and <sorted list of words> are the original words for that problem instance, but sorted. Two successive lines should be separated by a single blank line, but do not output any trailing blank line.

### Sample Input

```
3
3
beta alpha gamma
7
eastern bull cited another fizzy dubious gradient
6
this problem is not very hard
```

### Sample Output

```
Case 1: alpha beta gamma

Case 2: another bull cited dubious eastern fizzy gradient

Case 3: hard is not problem this very
```

## E. Alphacode

Alice and Bob need to send secret messages to each other and are discussing ways to encode their messages:

Alice: "Let's just use a very simple code: We'll assign 'A' the code word 1, 'B' will be 2, and so on down to 'Z' being assigned 26."

Bob: "That's a stupid code, Alice. Suppose I send you the word 'BEAN' encoded as 25114. You could decode that in many different ways!"

Alice: "Sure you could, but what words would you get? Other than 'BEAN', you'd get 'BEAAD', 'YAAD', 'YAN', 'YKD' and 'BEKD'. I think you would be able to figure out the correct decoding. And why would you send me the word 'BEAN' anyway?"

Bob: "OK, maybe that's a bad example, but I bet you that if you got a string of length 500 there would be tons of different decodings and with that many you would find at least two different ones that would make sense."

Alice: "How many different decodings?"

Bob: "Jillions!"

For some reason, Alice is still unconvinced by Bob's argument, so she requires a program that will determine how many decodings there can be for a given string using her code.

### Input

Input will consist of multiple input sets. Each set will consist of a single line of digits representing a valid encryption (for example, no line will begin with a 0). There will be no spaces between the digits. An input line of '0' will terminate the input and should not be processed

### Output

For each input set, output the number of possible decodings for the input string. All answers will be within the range of a long variable.

### Sample Input

```
25114
1111111111
3333333333
0
```

### Sample Output

```
6
89
1
```

## F. High with the Pie

The Pizazz Pizzeria prides itself in delivering pizzas to its customers as fast as possible. Unfortunately, due to cutbacks, they can afford to hire only one driver to do the deliveries. He will wait for 1 or more (up to 10) orders to be processed before he starts any deliveries. Needless to say, he would like to take the shortest route in delivering these goodies and returning to the pizzeria, even if it means passing the same location(s) or the pizzeria more than once on the way. He has commissioned you to write a program to help him.

### Input

Input will consist of multiple test cases. The first line will contain a single integer  $n$  indicating the number of orders to deliver, where  $1 \leq n \leq 10$ . After this will be  $n + 1$  lines each containing  $n + 1$  integers indicating the times to travel between the pizzeria (numbered 0) and the  $n$  locations (numbers 1 to  $n$ ). The  $j$ -th value on the  $i$ -th line indicates the time to go directly from location  $i$  to location  $j$  without visiting any other locations along the way. Note that there may be quicker ways to go from  $i$  to  $j$  via other locations, due to different speed limits, traffic lights, etc. Also, the time values may not be symmetric, i.e., the time to go directly from location  $i$  to  $j$  may not be the same as the time to go directly from location  $j$  to  $i$ . An input value of  $n = 0$  will terminate input.

### Output

For each test case, you should output a single number indicating the minimum time to deliver all of the pizzas and return to the pizzeria.

### Sample Input

```
3
0 1 10 10
1 0 1 2
10 1 0 10
10 2 10 0
0
```

### Sample Output

```
8
```