UW Madison's

# 2007 ACM-ICPC Individual Placement Test

October 7, 1:00-6:00pm, 1350 CS

**Overview:**
This test consists of seven coding problems and one written problem, which will be referred to by the following names (respective of order):

**maya, transport, euro, cheese, tables, phobos, land**

Solve the written problem and as many of the coding problems as you can. We have sorted these problems with respect to what we believe to be their relative difficulty, so it would be a good strategy (but by no means required) to start with problem “maya” and work your way through consecutively.

**Input/Output:**
Your programs should take input from standard in (i.e., the keyboard) and output to standard out (i.e., the terminal). As is standard practice for the ICPC, you may assume that the input strictly follows the description in the problems. It is your responsibility to ensure that your output **precisely** matches that described in the problems, or else risk your program being rejected with a “Presentation Error”.

**Problem Submission:**
To submit a program, send e-mail to sfdiehl@cs.wisc.edu and attach your source code. The subject line should contain only the problem name, **prob**, that you are submitting, and the attached source code should be named **prob**.{c|cpp|java}. For example, if you are using C++ and submitting the problem “maya”, the file should be named “maya.cpp”. You will receive the results of your submission as soon as possible via e-mail.

**Clarifications:**
As in the ICPC, you may submit clarification requests as well. They should be sent to sfdiehl@cs.wisc.edu, with a subject of “Clarification-**prob**”, where **prob** is the name of the problem you wish to be clarified. Replace **prob** with “general” if there is an issue with the contest as a whole. Accepted clarification requests will be answered to all those taking the test via e-mail. Experience of previous years learns that most clarification requests are rejected, and receive a simple response such as “Read the problem description”.

**Printing:**
You may print to the printer at any time during the test.

**Written Section:**
At the end of this packet there is a written section. We encourage you to spend the last half hour working on this section. Please describe an algorithmic solution to the problem at the end of the packet but do not write or submit code for this problem.

We would also like you to write down the main idea behind the solution to any of the problems for which you have not had a program accepted. Please be concise, using at most a few sentences within the space provided. We will take these partial solutions into account along with your ranking and other factors when composing teams.

**After the Test:**
The proctor will announce when time is up. Please stop working at this time and take a moment to fill out the form on the back of the written section and turn it in to the proctor (you may keep the problems). You are invited to join us for pizza and soda after the test in 1325 CS.

# 300 Maya Calendar

During his last sabbatical, professor M. A. Ya made a surprising discovery about the old Maya calendar. From an old knotted message, professor discovered that the Maya civilization used a 365 day long year, called *Haab*, which had 19 months. Each of the first 18 months was 20 days long, and the names of the months were *pop, no, zip, zotz, tzec, xul, yoxkin, mol, chen, yax, zac, ceh, mac, kankin, muan, pax, koyab, cumhu*. Instead of having names, the days of the months were denoted by numbers starting from 0 to 19. The last month of Haab was called *uayet* and had 5 days denoted by numbers 0, 1, 2, 3, 4. The Maya believed that this month was unlucky, the court of justice was not in session, the trade stopped, people did not even sweep the floor.

For religious purposes, the Maya used another calendar in which the year was called *Tzolkin* (holly year). The year was divided into thirteen periods, each 20 days long. Each day was denoted by a pair consisting of a number and the name of the day. They used 20 names: *imix, ik, akbal, kan, chicchan, cimi, manik, lamat, muluk, ok, chuen, eb, ben, ix, mem, cib, caban, eznab, canac, ahau* and 13 numbers; both in cycles.

Notice that each day has an unambiguous description. For example, at the beginning of the year the days were described as follows:

*1 imix, 2 ik, 3 akbal, 4 kan, 5 chicchan, 6 cimi, 7 manik, 8 lamat, 9 muluk, 10 ok, 11 chuen, 12 eb, 13 ben, 1 ix, 2 mem, 3 cib, 4 caban, 5 eznab, 6 canac, 7 ahau*, and again in the next period *8 imix, 9 ik, 10 akbal...*

Years (both Haab and Tzolkin) were denoted by numbers 0, 1, ..., where the number 0 was the beginning of the world. Thus, the first day was:

Haab: `0. pop 0`
Tzolkin: `1 imix 0`

Help professor M. A. Ya and write a program for him to convert the dates from the Haab calendar to the Tzolkin calendar.

## Input

The date in Haab is given in the following format:

*NumberOfTheDay. Month Year*

The first line of the input file contains the number of the input dates in the file. The next $n$ lines contain $n$ dates in the Haab calendar format, each in separate line. The year is smaller then 5000.

## Output

The date in Tzolkin should be in the following format:

*Number NameOfTheDay Year*

The first line of the output file contains the number of the output dates. In the next $n$ lines, there are dates in the Tzolkin calendar format, in the order corresponding to the input dates.

## Sample Input

```
3
10. zac 0
0. pop 0
10. zac 1995
```

## Sample Output

```
3
3 chuen 0
1 imix 0
9 cimi 2801
```

## 301 Transportation

Ruratania is just entering capitalism and is establishing new enterprising activities in many fields including transport. The transportation company TransRuratania is starting a new express train from city A to city B with several stops in the stations on the way. The stations are successively numbered, city A station has number 0, city B station number $m$. The company runs an experiment in order to improve passenger transportation capacity and thus to increase its earnings. The train has a maximum capacity $n$ passengers. The price of the train ticket is equal to the number of stops (stations) between the starting station and the destination station (including the destination station). Before the train starts its route from the city A, ticket orders are collected from all onroute stations. The ticket order from the station S means all reservations of tickets from S to a fixed destination station. In case the company cannot accept all orders because of the passenger capacity limitations, its rejection policy is that it either completely accept or completely reject single orders from single stations.

Write a program which for the given list of orders from single stations on the way from A to B determines the biggest possible total earning of the TransRuratania company. The earning from one accepted order is the product of the number of passengers included in the order and the price of their train tickets. The total earning is the sum of the earnings from all accepted orders.

### Input

The input file is divided into blocks. The first line in each block contains three integers: passenger capacity $n$ of the train, the number of the city B station and the number of ticket orders from all stations. The next lines contain the ticket orders. Each ticket order consists of three integers: starting station, destination station, number of passengers. In one block there can be maximum 22 orders. The number of the city B station will be at most 7. The block where all three numbers in the first line are equal to zero denotes the end of the input file.

### Output

The output file consists of lines corresponding to the blocks of the input file except the terminating block. Each such line contains the biggest possible total earning.

### Sample Input

```
10 3 4
0 2 1
1 3 5
1 2 7
2 3 10
10 5 4
3 5 10
2 4 9
0 2 5
2 5 8
0 0 0
```

## Sample Output

```
19
34
```

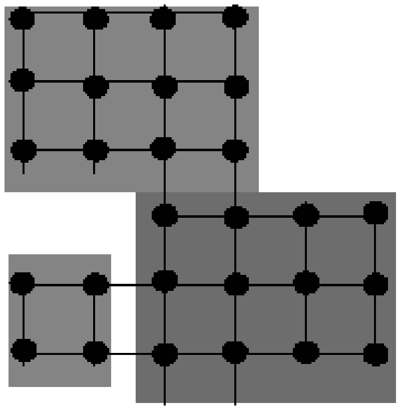
| | # 2724 - Eurodiffusion<br>World Finals - Beverly Hills - 2002/2003 | | | | |
|---|---|---|---|---|---|
| | PDF | PostScript | Submit | | Ranking |

On January 1, 2002, twelve European countries abandoned their national currency for a new currency, the euro. No more francs, marks, lires, guldens, kroner,... only euros, all over the eurozone. The same banknotes are used in all countries. And the same coins? Well, not quite. Each country has limited freedom to create its own euro coins:

> ``Every euro coin carries a common European face. On the obverse, member states decorate the coins with their own motif. No matter which motif is on the coin, it can be used anywhere in the 12 Member States. For example, a French citizen is able to buy a hot dog in Berlin using a euro coin with the imprint of the King of Spain." (source: http://europa.eu.int/euro/html/entry.html)

On January 1, 2002, the only euro coins available in Paris were French coins. Soon the first non-French coins appeared in Paris. Eventually, one may expect all types of coins to be evenly distributed over the twelve participating countries. (Actually this will not be true. All countries continue minting and distributing coins with their own motifs. So even in a stable situation, there should be an excess of German coins in Berlin.) So, how long will it be before the first Finnish or Irish coins are in circulation in the south of Italy? How long will it be before coins of each motif are available everywhere?

You must write a program to simulate the dissemination of euro coins throughout Europe, using a highly simplified model. Restrict your attention to a single euro denomination. Represent European cities as points in a rectangular grid. Each city may have up to 4 neighbors (one to the north, east, south and west). Each city belongs to a country, and a country is a rectangular part of the plane. The figure below shows a map with 3 countries and 28 cities. The graph of countries is connected, but countries may border holes that represent seas, or non-euro countries such as Switzerland or Denmark. Initially, each city has one million (1000000) coins in its country's motif. Every day a representative portion of coins, based on the city's beginning day balance, is transported to each neighbor of the city. A representative portion is defined as one coin for every full 1000 coins of a motif.



A city is *complete* when at least one coin of each motif is present in that city. A country is *complete* when all of its cities are complete. Your program must determine the time required for each country to become complete.

## Input

The input consists of several test cases. The first line of each test case is the number of countries ( $1 \leq c \leq 20$). The next $c$ lines describe each country. The country description has the format: *name* $x_l$ $y_l$ $x_h$ $y_h$, where *name* is a single word with at most 25 characters; $x_l$, $y_l$ are the lower left city coordinates of that country (most southwestward city ) and $x_h$, $y_h$ are the upper right city coordinates of that country (most northeastward city). $1 \leq x_l \leq xh \leq 10$ and $1 \leq y_l \leq y_h \leq 10$.

The last case in the input is followed by a single zero.

## Output

For each test case, print a line indicating the case number, followed by a line for each country with the country name and number of days for that country to become complete. Order the countries by days to completion. If two countries have identical days to completion, order them alphabetically by name.

Use the output format shown in the example.

## Sample Input

```
3
France  1 4 4 6
Spain          3 1 6 3
Portugal    1 1 2 2
1
Luxembourg  1 1 1 1
2
Netherlands 1 3 2 4
Belgium     1 1 2 2
0
```

## Sample Output

```
Case Number 1
   Spain   382
   Portugal   416
   France   1325
Case Number 2
   Luxembourg   0
Case Number 3
   Belgium   2
   Netherlands   2
```

---

# Problem B
## Say Cheese
## Input: cheese.in

Once upon a time, in a giant piece of cheese, there lived a cheese mite named Amelia Cheese Mite. Amelia should have been truly happy because she was surrounded by more delicious cheese than she could ever eat. Nevertheless, she felt that something was missing from her life.

One morning, her dreams about cheese were interrupted by a noise she had never heard before. But she immediately realized what it was — the sound of a male cheese mite, gnawing in the same piece of cheese! (Determining the gender of a cheese mite just by the sound of its gnawing is by no means easy, but all cheese mites can do it. That's because their parents could.)

Nothing could stop Amelia now. She had to meet that other mite as soon as possible. Therefore she had to find the fastest way to get to the other mite. Amelia can gnaw through one millimeter of cheese in ten seconds. But it turns out that the direct way to the other mite might not be the fastest one. The cheese that Amelia lives in is full of holes. These holes, which are bubbles of air trapped in the cheese, are spherical for the most part. But occasionally these spherical holes overlap, creating compound holes of all kinds of shapes. Passing through a hole in the cheese takes Amelia essentially zero time, since she can fly from one end to the other instantly. So it might be useful to travel through holes to get to the other mite quickly.

For this problem, you have to write a program that, given the locations of both mites and the holes in the cheese, determines the minimal time it takes Amelia to reach the other mite. For the purposes of this problem, you can assume that the cheese is infinitely large. This is because the cheese is so large that it never pays for Amelia to leave the cheese to reach the other mite (especially since cheese-mite eaters might eat her). You can also assume that the other mite is eagerly anticipating Amelia's arrival and will not move while Amelia is underway.

## Input

The input file contains descriptions of several cheese mite test cases. Each test case starts with a line containing a single integer $n$ ($0 \le n \le 100$), the number of holes in the cheese. This is followed by $n$ lines containing four integers $x_i, y_i, z_i, r_i$ each. These describe the centers $(x_i, y_i, z_i)$ and radii $r_i$ ($r_i > 0$) of the holes. All values here (and in the following) are given in millimeters.

The description concludes with two lines containing three integers each. The first line contains the values $x_A, y_A, z_A$, giving Amelia's position in the cheese, the second line containing $x_O, y_O, z_O$, gives the position of the other mite.

The input file is terminated by a line containing the number –1.

## Output

For each test case, print one line of output, following the format of the sample output. First print the number of the test case (starting with 1). Then print the minimum time in seconds it takes Amelia to reach the other mite, rounded to the closest integer. The input will be such that the rounding is unambiguous.

**Sample Input**

```
1
20 20 20 1
0 0 0
0 0 10
1
5 0 0 4
0 0 0
10 0 0
-1
```

**Output for the Sample Input**

```
Cheese 1: Travel time = 100 sec
Cheese 2: Travel time = 20 sec
```

The 26th Annual

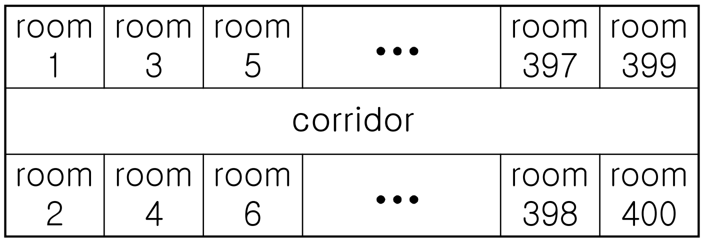
# ACM International Collegiate Programming Contest ASIA Regional - Taejon

## Problem F

### Moving Tables

### Input: table.in

The famous ACM (Advanced Computer Maker) Company has rented a floor of a building whose shape is in the following figure.



The floor has 200 rooms each on the north side and south side along the corridor. Recently the Company made a plan to reform its system. The reform includes moving a lot of tables between rooms. Because the corridor is narrow and all the tables are big, only one table can pass through the corridor. Some plan is needed to make the moving efficient. The manager figured out the following plan: Moving a table from a room to another room can be done within 10 minutes. When moving a table from room *i* to room *j*, the part of the corridor between the front of room *i* and the front of room *j* is used. So, during each 10 minutes, several moving between two rooms not sharing the same part of the corridor will be done simultaneously. To make it clear the manager illustrated the possible cases and impossible cases of simultaneous moving.

| | Table moving | Reason |
|---|---|---|
| Possible | ( room 30 to room 50) and (room 60 to room 90) | no part of corridor is shared |
| | (room 11 to room 12) and (room 14 to room 13) | no part of corridor is shared |
| Impossible | (room 20 to room 40) and (room 31 to room 80) | corridor in front of room 31 to room 40 is shared |
| | (room 1 to room 4) and (room 3 to room 6) | corridor in front of room 3 is shared |
| | (room 2 to room 8) and (room 7 to room 10) | corridor in front of room 7 is shared |

For each room, at most one table will be either moved in or moved out. Now, the manager seeks out a method to minimize the time to move all the tables. Your job is to write a program to solve the manager's problem.

### Input

The input consists of $T$ test cases. The number of test cases $(T)$ is given in the first line of the input file. Each test case begins with a line containing an integer $N$, $1 \le N \le 200$, that represents the number of tables to move. Each of the following $N$ lines contains two positive integers *s* and *t*, representing that a table is to move from room number *s* to room number *t* (each room number appears at most once in the *N* lines). From the $N+3$-rd line, the remaining test cases are listed in the same manner as above.

## Output

The output should contain the minimum time in minutes to complete the moving, one per line.

**Sample Input (table.in)**

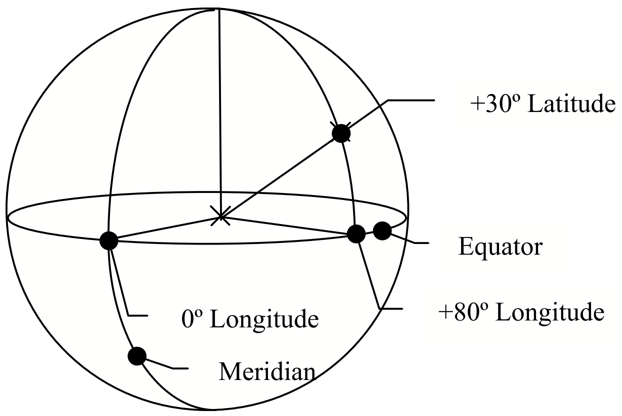
```
3
4
10 20
30 40
50 60
70 80
2
1 3
2 200
3
10 100
20 80
30 50
```

**Output for the Sample Input**

```
10
20
30
```

# Problem 5: Communication Planning for Phobos

Life has been found on Phobos, one of the satellites of Mars! Unfortunately, the life forms there aren't quite as advanced as those on Earth, and they don't have modern communications (at least by Earth standards). The Advanced Communication Management Company (ACM) has decided to build a central office and connect the Phobosians' homes for communication (telephone, television, Internet, and so forth). They naturally want to minimize their capital outlay in this effort, and they need to decide how to lay fiber optic cable (essentially on the surface) so the smallest amount is used. Since ACM uses digital broadband technology, it is only necessary that there be a cable path that connects every subscriber and the central office. That is, there does not necessarily need to be a separate cable from the central office to each subscriber's home.



We know the precise location of each Phobosian's home and the planned ACM central office on the surface. These are given using longitude and latitude. Longitude is measured from an arbitrary meridian on the surface of Phobos, and has values in the range –180 degrees to +180 degrees. Latitude is measured from the equator, and has values in the range –90 degrees to +90 degrees. For planning purposes we assume Phobos is perfectly spherical, exactly 16.7 miles in diameter. The figure to the left illustrates one possible location (+80° longitude, +30° latitude).

**INPUT**
There will be one or more sets of input data. Each set will contain, in order, an integer *N* no larger than 100, but at least 2, followed by *N* pairs of real numbers, each pair giving the unique longitude and latitude, in degrees, of a Phobosian's home or the central office. A single integer zero will follow the last data set.

**OUTPUT**
For each input data set print a single line containing the data set number (1, 2, …) and the number of miles of cable required to connect all the Phobosian's homes and the central office; show two fractional digits in the distance.

**SAMPLE INPUT**

```
3
0 0    0 90    0 -90

3
0 0    0 90    90 0

3
0 0    90 0    45 0

6
-10 10   -10 -10   0 0   90 0   80 20   100 -10

0
```

**EXPECTED OUTPUT**

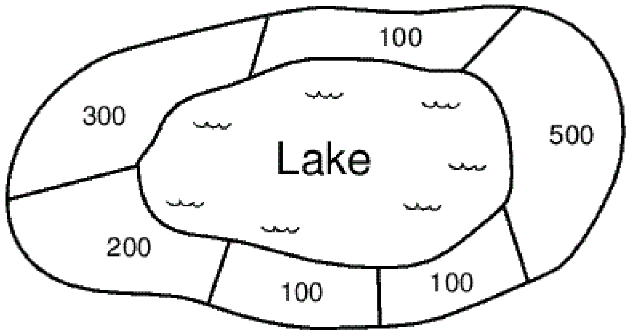
```
Case 1: 26.23 miles
Case 2: 26.23 miles
Case 3: 13.12 miles
Case 4: 21.16 miles
```

# 3151 - Land Division Tax

Latin America - South America - 2004/2005

| PDF | PostScript | Submit | | Ranking |
|---|---|---|---|---|

International Concrete Projects Company (ICPC) is a construction company which specializes in building houses for the high-end market. ICPC is planning a housing development for new homes around a lake. The houses will be built in lots of different sizes, but all lots will be on the lake shore. Additionally, every lot will have exactly two neighbors in the housing development: one to the left and one to the right.



Development plan indicating the sizes of the lots (in units of area) in the new housing development.

ICPC owns the land around the lake and needs to divide it into lots according to the housing development plan. However, the County Council has a curious regulation regarding land tax, intended to discourage the creation of small lots:

1. land can only be divided using a sequence of land divisions;
2. a land division is an operation that divides one piece of land into two pieces of land; and
3. for each land division, a land division tax must be paid.

Denoting by A the area of the largest resulting part of the division, the value of the land division tax is $A \times F$, where F is the division tax factor set yearly by the County Council. Note that due to (2), in order to divide a piece of land into N lots, N - 1 land divisions must be performed, and therefore N - 1 payments must be made to the County Council.

For example, considering the figure above, if the division tax factor is 2.5 and the first land division separates the lot of 500 units of area from the other lots, the land division tax to be paid for this first division is $2.5 \times (300 + 200 + 100 + 100 + 100)$. If the next land division separates the lot of 300 units together with its neighbor lot of 100 units, from the set of the remaining lots, an additional $2.5 \times (300 + 100)$ must be paid in taxes, and so on. Note also that some land divisions are not possible, due to (2). For example, after the first land division mentioned above, it is not possible to make a land division to separate the lot of 300 units together with the lot of 200 units from the remaining three lots, because more than two parts would result from that operation.

Given the areas of all lots around the lake and the current value of the division tax factor, you must write a program to determine the smallest total land division tax that should be paid to divide the land according to the housing development plan.

## Input

The input contains several test cases. The first line of a test case contains an integer N and a real F, indicating respectively the number of lots ($1 \le N \le 200$) and the land division tax factor (with precision of two decimal digits, $0 < F \le 5.00$). The second line of a test case contains N integers $X_i$, representing the areas of contiguous lots in the development plan ($0 < X_i \le 500$, for $1 \le i \le N$); furthermore, $X_k$ is neighbour to $X_{k+1}$ for $1 \le k \le N - 1$, and $X_N$ is neighbour to $X_1$. The end of input is indicated by $N = F = 0$.

## Output

For each test case in the input your program must produce a single line of output, containing the minimum total land division tax, as a real number with precision of two decimal digits.

## Sample Input

```
4 1.50
```

```
2 1 4 1
6 2.50
300 100 500 100 100 200
0 0
```

## Sample Output

```
13.50
4500.00
```

---

**Written Section:**

For problem "Chandelier" (on the following page), please write a detailed description of an efficient algorithm to solve this problem. Do not write or submit code. Be as clear and as concise as possible.

Use the space below to briefly write down the basic idea behind the solution to any problems for which you did not get a program accepted.

**maya:**

**transport:**

**cheese:**

**phobos:**

**euro:**

**land:**

**tables:**

**Information Form:**

Name: ____________________

CS Login: _________________

Student status (e.g., Junior, first year grad student): _________________

Year of birth:____________ , Year starting college:_________________

Which of C/C++/Java do you prefer? ____. Please indicate your proficiency in each language:

What classes have you taken (or are you taking) which are relevant to the ICPC?

What do you feel are your strengths with respect to the ICPC?

Are you able to attend the world finals in Banff, Canada, April 6-10, 2008 (and obtain a Visa or passport as necessary)?

How many ICPC regionals have you participated in? ____ How many ICPC world finals? ____

If your team progresses to the world finals, how many hours per week could you commit to practicing? _____

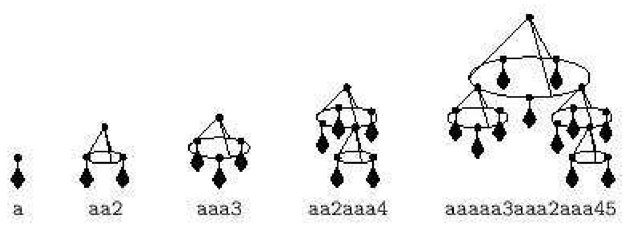
Is there anything else we should know about you?

# 3215 - Chandelier

Europe - Northeastern Europe & Russian Republic - 2004/2005

## ***DO NOT CODE THIS PROBLEM***

Lamps-O-Matic company assembles very large chandeliers. A chandelier consists of multiple levels. On the first level crystal pendants are attached to the rings. Assembled rings and new pendants are attached to the rings of the next level, and so on. At the end there is a single large ring -- the complete chandelier with multiple smaller rings and pendants hanging from it. A special-purpose robot assembles chandeliers. It has a supply of crystal pendants and empty rings, and a stack to store elements of a chandelier during assembly. Initially the stack is empty. Robot executes a list of commands to assemble a chandelier.



On command ``a" robot takes a new crystal pendant and places it on the top of the stack. On command ``1" to ``9" robot takes the corresponding number of items from the top of the stack and consecutively attaches them to the new ring. The newly assembled ring is then placed on the top of the stack. At the end of the program there is a single item on the stack -- the complete chandelier. Unfortunately, for some programs it turns out that the stack during their execution needs to store too many items at some moments. Your task is to optimize the given program, so that the overall design of the respective chandelier remains the same, but the maximal number of items on the stack during the execution is minimal. A pendant or any complex multi-level assembled ring count as a single item of the stack. The design of a chandelier is considered to be the same if each ring contains the same items in the same order. Since rings are circular it does not matter what item is on the top of the stack when the robot receives a command to assemble a new ring, but the relative order of the items on the stack is important. For example, if the robot receives command ``4" when items $\langle i_1, i_2, i_3, i_4 \rangle$ are on the top of the stack in this order ($i_1$ being the topmost), then the same ring is also assembled if these items are arranged on the stack in the following ways: $\langle i_2, i_3, i_4, i_1 \rangle$, or $\langle i_3, i_4, i_1, i_2 \rangle$, or $\langle i_4, i_1, i_2, i_3 \rangle$.

## Input

Input file contains several test cases. Each of them consists of a single line with a valid program for the robot. The program consists of at most 10 000 characters.

## Output

For each test case, print two output lines. On the first line write the minimal required stack capacity (number of items it can hold) to assemble the chandelier. On the second line write some program for the assembly robot that uses stack of this capacity and results in the same chandelier.

## Sample Input

```
aaaaa3aaa2aaa45
```

## Sample Output

```
6
aaa3aaa2aaa4aa5
```