مرکز مطالعات پارت
لذت کشف آینده

دانشگاه فردوسی مشهد

انجمن علمی گروه کامپیوتر

# مسابقات برنامه‌نویسی دانشگاه فردوسی مشهد

# FCPC

## (طرح آموزشی)

**ویژه دانشجویان فردوسی**

**۲۴ اسفند ۱۴۰۲**

## توضیحات:

- مسابقه شامل ۸ سوال در ۱۲ صفحه و زمان پاسخگویی ۳ ساعت است.
- کامپایلرهای موجود C, CPP, Java, Python هستند.
- ورودی‌ها و خروجی‌های تمام سوالات، ورودی/خروجی استاندارد است.

| Question | Title |
|---|---|
| A | FCPC |
| B | The Grand Gathering |
| C | FUM on Fire |
| D | World Cup Draw |
| E | Bank Card Verifier |
| F | Nader Shah |
| G | Congestion Charging Zone |
| H | Cinema |

# Problem A: FCPC

The FUM ChamPions Cup (FCPC), the most prestigious football league in Iran, is reaching its end, and people are eagerly waiting for the finals, which happened to be between the two most popular Iranian teams, Persepolis and Esteghlal.

The FCPC finals consist of two matches, with each team competing as the home team in one match. The winning team is determined by aggregate score, the sum of the scores of the two matches. For example, if the scores of the two matches are Persepolis 6–0 Esteghlal in the first match, and Esteghlal 3–1 Persepolis in the second match, then the aggregate score will be Persepolis 7–3 Esteghlal, meaning that Persepolis is the winner. If aggregates are equal, the away goals rule is used to determine the winner, in which case the winner is the team that scored the most goals in the match it played away from home. If the result is still equal, a penalty shootout is required.

Saeed and Yaser have made a bet on the result of the match, and now each one of them is trying to figure out various scenarios in which his favorite team wins the finals and the other's loses. Since they are both professional programmers and have important things to do, they ask you to write a program that takes the number of goals in the two matches as input, and determines which team is the winner based on the aggregate scores and the away goals rule. If it cannot be determined from this information, the program should declare that the match would go to penalty kicks.

## Input (standard input)

The first line of the input contains two space-separated integers $p_1$ and $s_1$, where $p_1$ and $s_1$ are the number of goals scored by Persepolis and Esteghlal, respectively, in the first match in which Persepolis is the home team. The second line contains two space-separated integers $s_2$ and $p_2$, where $s_2$ and $p_2$ are the number of goals scored by Esteghlal and Persepolis, respectively, in the second match in which Esteghlal is the home team. All input integers are between 0 and 20, inclusively.

## Output (standard output)

In the output, print the name of the winning team, either Persepolis or Esteghlal, if the winner can be determined by the aggregate scores and the away goals rule. Otherwise, print Penalty

## Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| `3 0`<br>`2 1` | `Persepolis` |

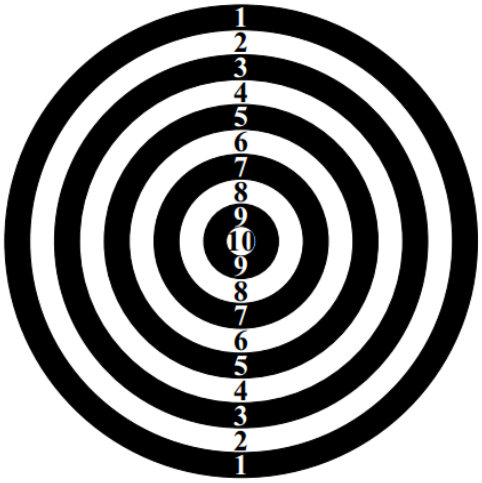
| Standard Input | Standard Output |
|---|---|
| `3 1`<br>`2 0` | `Esteghlal` |

| Standard Input | Standard Output |
|---|---|
| `2 0`<br>`2 0` | `Penalty` |

## Problem B: The Grand Gathering

The highly anticipated first FCPC-Selection Division is finally going to be held after the new year holidays! The Executive committee is planning a post-contest gathering event to celebrate the long awaited contest. Some entertaining programs like phantomime, fooshball, fench, yarpelle, and several board games are part of the event. You are going to set up a dart game in this gathering. As a techie organizing a game for techies, you would rather use a smart screen and write a program to calculate the scores instead of hanging a traditional dartboard and scoring the shots manually. Your program must get calculated based on its distance from the center of the dartboard (point (0, 0)). If the distance is d millimeters, the score is calculated based on the following table:

| **Condition** | **Score** |
|---|---|
| $d \leqslant 10$ | 10 |
| $10 < d \leqslant 30$ | 9 |
| $30 < d \leqslant 50$ | 8 |
| $50 < d \leqslant 70$ | 7 |
| $70 < d \leqslant 90$ | 6 |
| $90 < d \leqslant 110$ | 5 |
| $110 < d \leqslant 130$ | 4 |
| $130 < d \leqslant 150$ | 3 |
| $150 < d \leqslant 170$ | 2 |
| $170 < d \leqslant 190$ | 1 |
| $190 < d$ | 0 |



### Input (standard input)

The first line of the input contains a single integer $N$ as the number of dart shots for a player ($1 \leqslant N \leqslant 100$). Each of the next $N$ lines contains two space-separated integers as the coordinates $(x, y)$ of a dart shot. The coordinates are in millimeters and their absolute values will not be greater than 300.

### Output (standard output)

Print a single line containing the total score of the player

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| `2`<br>`4 7`<br>`-31 -5` | `18` |

| Standard Input | Standard Output |
|---|---|
| `3`<br>`12 -16`<br>`-180 100`<br>`152 10` | `11` |

# Problem C: FUM on fire

The campus of Ferdowsi University of Mashhad is on fire as a result of the dangerous cracker games and fireworks of Chaharshanbe Suri. The fire has already consumed several buildings and is rapidly spreading to other areas. In the meantime, Amirhosein finds himself as the sole survivor amidst the fiery inferno. With no time to spare, he must make a daring attempt to reach the only helicopter on campus in order to save himself from the impending disaster. Suppose the campus of FUM is represented by an n×m grid. At the initial time, some grid cells are on fire. If a cell catches fire at time x, all its 8 vertex-neighboring cells will catch fire at time x + k. If a cell catches fire, it will be on fire forever. At the initial time, Amirhosein stands at cell s and the helicopter is located at cell t. At any time x, Amirhosein can move from its current cell to one of four edge-neighboring cells, located at the left, right, top, or bottom of its current cell if that cell is not on fire at time x + 1. Note that each move takes one second. Your task is to write a program to find the shortest path from s to t avoiding fire.

## Input (standard input)

There are multiple test cases in the input. The first line of each test case contains three positive integers $n$, $m$ and $k$ ($1 \leqslant n; m; k \leqslant 100$), where $n$ and $m$ indicate the size of the test case grid $n \times m$, and $k$ denotes the growth rate of fire. The next $n$ lines, each contains a string of length $m$, where the $j$th character of the $i$th line represents the cell $(i, j)$ of the grid. Cells which are on fire at time 0, are presented by character “f”. There may exist no “f” in the test case. The helicopter and Amirhosein are located at cells presented by “t” and “s”, respectively. Other cells are filled by “-” characters. The input terminates with a line containing “0 0 0” which should not be processed.

## Output (standard output)

For each test case, output a line containing the shortest time to reach $t$ from $s$ avoiding fire. If it is impossible to reach $t$ from $s$, write “Impossible” in the output.

## Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 7 7 2<br>f------<br>-f---f-<br>----f--<br>-------<br>------f<br>---s---<br>t----f-<br>3 4 1<br>t--f<br>--s-<br>----<br>2 2 1<br>st<br>f-<br>2 2 2<br>st<br>f-<br>0 0 0 | 4<br>Impossible<br>Impossible<br>1 |

## Problem D: World Cup Draw

The draw for the 2018 FIFA World Cup took place on Friday, December 1 at the State Kremlin Palace in Moscow. The FIFA released the draw procedure a few months ago on the official FIFA website, explained below. The 32 qualified finalists are first distributed into four seeding pots based on the FIFA ranking for October 2017. Pot 1 contains the host Russia and the seven highest ranking teams; the next eight highest ranked teams in Pot 2, and so on. Then, teams are drawn into eight groups of four, which are labeled from *A* to *H*. The pots are emptied into groups in order from Pot 1 through Pot 4. The draw must satisfy the following two rules:

(i) no teams from the same pot can be drawn into the same group.

(ii) with the exception of UEFA, which has more qualified teams (14) than the groups (8), no teams from the same confederation can be drawn in the same group. Moreover, at most two teams from UEFA can be drawn in the same group.

Order groups alphabetically from *A* to *H* with *A* and *H* respectively being the leftmost and rightmost groups. At each step of the draw, the drawn team *x* from Pot *i* is placed in the first group from left (starting from group *A*) not violating rules (i) and (ii) and being possible to distribute the remaining teams (not drawn teams so far) of Pot *i* in the next steps without violating rules (i) and (ii). Computer scientists have assured FIFA that it is always possible to distribute teams of Pot *i* into groups satisfying rules (i) and (ii), regardless of how the other teams in Pot 1 through Pot *i* - 1 are distributed into groups.

The table below shows the pots. In this table, *x* (*r*, *c*) means that team *x* belongs to confederation c, and its rank in the FIFA ranking is *r*. You are to write a program to simulate the draw and report the groups for the given draw order for each pot.

| Seeding Pot 1 | Seeding Pot 2 | Seeding Pot 3 | Seeding Pot 4 |
|---|---|---|---|
| Russia (65, UEFA) | Spain (8, UEFA) | Denmark (19, UEFA) | Serbia (38, UEFA) |
| Germany (1, UEFA) | Peru (10, CONMEBOL) | Iceland (21, UEFA) | Nigeria (41, CAF) |
| Brazil (2, CONMEBOL) | Switzerland (11, UEFA) | Costa Rica (22, CONCACAF) | Australia (43, AFC) |
| Portugal (3, UEFA) | England (12, UEFA) | Sweden (25, UEFA) | Japan (44, AFC) |
| Argentina (4, CONMEBOL) | Colombia (13, CONMEBOL) | Tunisia (28, CAF) | Morocco (48, CAF) |
| Belgium (5, UEFA) | Mexico (16, CONCACAF) | Egypt (30, CAF) | Panama (49, CONCACAF) |
| Poland (6, UEFA) | Uruguay ( 17, CONMEBOL) | Senegal (32, CAF) | South Korea (62, AFC) |
| France (7, UEFA) | Croatia (18, UEFA) | Iran (34, AFC) | Saudi Arabia (63, AFC) |

### Input (Standard Input)

There are multiple test cases in the input. Each test case consists of 4 lines. The *i*th line presents all 8 team names (as written in the table) in Pot *i* in the draw order (from left to right). Team names are separated by ”,” and there may exist a space before or after team names. In all test cases, Russia is the first drawn country in Pot 1 due to the old tradition of placing the host country in group *A*. The input terminates with “End” that should not be processed .

## Output (standard output)

For each test case, simulate the draw based on the given draw order, and compute the weight of each group which is the summation of team ranks in that group. For each group, print the group name (an uppercase letter) and its weight separated by a space in one line. This must be done in the increasing order of the weights from the strongest group (the group with the minimum weight) to the weakest group (the group with the maximum weight). In the case of a tie, print based on the alphabetical order of group names.

## Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| Russia, Germany, Brazil, Portugal, Argentina, Belgium, Poland, France<br>Spain, Peru, Switzerland, England, Colombia, Mexico, Uruguay, Croatia<br>Denmark, Iceland, Costa Rica, Sweden, Tunisia, Egypt, Senegal, Iran<br>Serbia, Nigeria, Australia, Japan, Morocco, Panama, South Korea, Saudi Arabia<br>Russia, Germany, Brazil, Portugal, Argentina, Belgium, Poland, France<br>Spain, Peru, Switzerland, England, Colombia, Mexico, Uruguay, Croatia<br>Denmark, Iceland, Costa Rica, Sweden, Tunisia, Egypt, Senegal, Iran<br>Serbia, Nigeria, Morocco, Panama, Australia, Japan, South Korea, Saudi Arabia<br>End | B 73<br>C 78<br>E 83<br>D 92<br>H 107<br>F 110<br>G 118<br>A 136<br>C 77<br>B 78<br>E 83<br>D 87<br>H 108<br>F 110<br>G 118<br>A 136 |

## Problem E: Bank Card Verifier

Part software group, is looking for creative software developers. A group of applicants are attending an interview, and the company wants to select the fastest developer who can code simple rules accurately. As a test, all applicants should quickly develop a bank card verifier that determines whether a payment card number is valid or not.

All payment card numbers are 16 digits long. The leftmost 6 digits represent a unique identification number for the bank who has issued the card. The next 2 digits determine the type of the card (e.g., debit, credit, gift). Digits 9 to 15 are the serial number of the card, and the last digit is used as a control digit to verify whether the card number is valid. Hence, if somebody enters the card number incorrectly, there is a high chance that a payment software can easily determine it.

For a valid card number, the last digit is selected in such a way that the following algorithm passes:

1. Label all digits from left to right by 1 to 16.
2. Multiply each odd-labeled digit by 2.
3. If the result for any digit is greater than 9, subtract 9 from it
4. Sum the results of the previous step, and add to it the sum of all even-labeled digits.
5. If the result is a multiple of 10, the card number is valid; otherwise, it is invalid.

Your task is to read several card numbers from the input, and determine whether each one is a valid card number or not.

### Input (standard input)

There are multiple test cases in the input. Each test is given in one line consisting of four space-separated 4-digit strings. The leftmost digit of the given card number is guaranteed to be non-zero. The input terminates with a line containing “0000 0000 0000 0000” that should not be processed.

### Output (standard output)

For each test case, output a line containing “Yes” or “No” depending on whether the card number is valid or not, respectively.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 6104 3376 7866 1545<br>6104 3376 7866 1546<br>5022 2910 0140 7954<br>0000 0000 0000 0000 | Yes<br>No<br>Yes |

## Problem F: Nader Shah

Nader Shah Afshar was one of the most powerful rulers in Iran. He won many battles, such as the battle of Herat, Mihmandust, and Kirkuk. During his ruling, Mashhad was Iran’s capital. Each year, he planned an attack on a new country, and annexed it to Iran’s territory. As Nader Shah’s past victories were known to everyone, the country under attack was surrendering the battle without any combat. Therefore, after some years, Iran’s territory was expanded to *n* new countries.

There was a connected road network between the countries. All roads were two-way passing. Nader Shah only attacked those countries which had at least one road to Iran’s territory at the moment. After capturing each country, Nader Shah chose exactly one road between the captured country and Iran’s territory at that moment, and put Iran flag on it at every one kilometer. In order to make costs as minimal as possible, he chose the shortest road as it needed less Iran flags (if there are more than one shortest road, he arbitrarily chose one of them). These roads were called Afshari roads. Obviously, by choosing the roads this way, there are exactly $k$ - 1 Afshari roads when Iran’s territory consists of $k$ countries.

Several years later, we have received a map of Iran’s territory at the end of Nader Shah’s life. The map has all the road network on it with their lengths in kilometers. Afshari roads are marked in the map. There are $n$ countries on the map, $m$ roads connecting them, and $n$ - 1 of those roads marked as Afshari. Each country is marked by a distinct number $1 \leqslant x \leqslant n$. We do not distinguish between the numbers and the countries.

Given the map described above, we want to know the possible country corresponding to Iran and the order by which Nader Shah captured countries. One possible answer can be shown as $a_1$, $a_2$, …, $a_n$ where $a_1$ corresponds to Iran, $a_2$ is the first country that Nader Shah captured, $a_3$ is the second country he captured, and so on. We want to know the lexicographically minimum possible answer. Array $x_1, x_2, \ldots , x_n$ is lexicographically smaller than $y_1, y_2, \ldots , y_n$ if and only if there exists a number $i$ $(1 \leqslant i \leqslant n)$ that $(x_1 = y_1) \land (x_2 = y_2) \land \ldots \land (x_{i-1} = y_{i-1})$ and $x_i < y_i$.

### Input (standard input)

The first line of the input contains two integers $n$ and $m$ $(1 \leqslant n \leqslant 1,000,\ n - 1 \leqslant m \leqslant 5,000)$ where $n$ and $m$ are the number of countries and the number of roads in the map (the road network), respectively. The next $m$ lines describe the road network; each line contains four integers $u$, $v$, $w$, and $r$ $(1 \leqslant u, v \leqslant n,\ 1 \leqslant w \leqslant 10^6;$ and $r \in \{0, 1\})$ which indicate there is a road between $u$ and $v$ with the length of $w$ kilometers, and $r$ denotes whether the road is Afshari or not. The road is Afshari if and only if $r = 1$. There is at most one road between two countries. It is guaranteed that there are exactly $n$ - 1 Afshari roads in the road network, and all $n$ countries are connected through these $n$ - 1 roads.

### Output (standard output)

If there is no answer to the problem, print Wrong Map!. Otherwise, write the lexicographically minimum answer to the problem.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 3 3<br>1 2 2 1<br>2 3 2 1<br>1 3 1 0 | Wrong Map! |

| Standard Input | Standard Output |
|---|---|
| 3 3<br>1 2 1 1<br>2 3 2 0<br>1 3 3 1 | 1 3 2 |

# Problem G: Congestion Charging Zone

Tehran municipality has set up a new charging method for the Congestion Charging Zone (CCZ) which controls the passage of vehicles in Tehran's high-congestion areas in the congestion period (CP) from 6:30 to 19:00. There are plate detection cameras inside or at the entrances of the CCZ recording vehicles seen at the CCZ. The table below summarizes the new charging method.

| The first time seen in the CP | The last time seen in the CP | Charge |
|---|---|---|
| 6:30 to 10:00 | 6:30 to 16:00 | 24000 |
| 6:30 to 10:00 | 16:01 to 19:00 | 36000 |
| 10:01 to 16:00 | 10:01 to 16:00 | 16800 |
| 10:01 to 19:00 | 16:01 to 19:00 | 24000 |

Note that the first time and the last time that a vehicle is seen in the CP may be the same. Write a program to compute the amount of charge of a given vehicle in a specific day.

## Input (standard input)

The first line of the input contains a positive integer $n$ ($1 \leqslant n \leqslant 100$) where $n$ is the number of records for a vehicle. Each of the next $n$ lines contains a time at which the vehicle is seen. Each time is of form <hour>:<minute>, where <hour> is an integer number between 0 and 23 (inclusive) and <minute> is formatted as an exactly two-digit number between 00 and 59 (inclusive).

## Output (standard output)

Print the charge to be paid by the owner of the vehicle in the output.

## Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 4<br>7:30<br>2:20<br>7:30<br>17:30 | 36000 |

| Standard Input | Standard Output |
|---|---|
| 1<br>12:13 | 16800 |

| Standard Input | Standard Output |
|---|---|
| 2<br>0:30<br>23:30 | 0 |

## Problem H: Cinema

The main movie theater of the city consists of a single auditorium with rows of comfortable padded seats. Surprisingly, the comfortness of seats are not necessarily equal. Precisely, each seat has its own comfort value which is a non-negative integer number. A seat is more comfortable than another seat if its comfort value is larger. It is only possible to enter a row from the left side of the auditorium. Assume the seats in a row are numbered 1 to $m$ from left to right. When a person enters a row, he/she always sits on the most comfortable seat which is free and accessible to him/her. If he/she sits at seat $i$, he/she blocks other persons coming later to sit on seats $i$+1 to $m$. If there are more than one free and accessible seat being the most comfortable, he/she sits on the leftmost one. The owner of the movie theater plans to improve the comfortness of some seats to have more audiences in the auditorium. Improving one unit in the comfortness of a seat costs some fix value. With the budget available, the owner knows the total improvement over all seats must not exceed a value $k$. Help the owner find the best way to improve the comfortness of seats by at most $k$ units in total to have the maximum number of audiences in the auditorium.

### Input (standard input)

The first line contains three non-negative integers $n$, $m$, and $k$ ($1 \leqslant n{\cdot}m \leqslant 3\times10^5$, $0 \leqslant k \leqslant 10^{12}$) which are the number of rows, the number of seats in each row, and the total comfortness that can be added to all seats. The next $n$ lines describe the comfort values of seats; each line contains $m$ non-negative integers not more than $10^6$ denoting the comfort values of seats from left to right for a row.

### Output (standard output)

Print a single line containing the maximum number of audiences.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 2 3 10<br>10 1 12<br>8 3 6 | 5 |

| Standard Input | Standard Output |
|---|---|
| 1 4 6<br>9 8 10 8 | 3 |

| Standard Input | Standard Output |
|---|---|
| 1 3 2<br>10 10 10 | 2 |