Concours interne de programmation
Télécom ParisTech

28 juin 2018

Contents

Problem setters

- Antoine Amarilli
- Étienne Borde
- Florian Brandner
- Bertrand Meyer

Special thanks to Pierre Senellart for setting up and maintaining the judging system.

Barème

Barème pour le cours INF280

Si vous êtes inscrit au cours INF280 en 2017–2018, votre solution pour chaque exercice sera évaluée : elle recevra la note maximale pour cet exercice si elle est acceptée par le juge en ligne, sinon elle recevra une note fractionnaire. Chaque exercice a le même poids dans le barème final.

Attention : vous devez soumettre votre solution, même partielle, sur le juge en ligne pour être corrigé.

Les soumissions incorrectes et le temps de soumission ne sont pas pris en compte dans la note.

Classement

Un classement de l'ensemble des participants sera établi à l'issue du concours. Il est indépendent du cours INF280. Il est basé uniquement sur les exercices pour lesquels votre code est accepté par le juge, et il suit les règles du SWERC.

1. Les participants sont classés en premier lieu par nombre décroissant d'exercices résolus.
2. Pour les participants qui ont résolu le même nombre d'exercices, les candidats sont classés par *temps de départage* croissant. Le temps de départage d'un candidat est la somme, pour chaque problème résolu par ce candidat, du temps entre le début du concours et la soumission de la première solution acceptée pour ce problème par ce candidat. Chaque soumission rejetée pour un problème finalement résolu par le candidat ajoute un malus de 20 minutes au temps de départage du candidat. Les soumissions rejetées sont sans effet si elles concernent un problème que le candidat n'a pas finalement résolu.

Les étudiants seront appelés, dans l'ordre du classement, à représenter Télécom au concours ACM-ICPC SWERC 2018 qui se déroulera le week-end du 1–2 décembre 2018 à Télécom. Les six premiers étudiants éligibles et volontaires seront retenus.

Problem A: It's a Jungle Out There

Time limit: 5 seconds

As more and more highways are cutting trough the wild forests and habitats of snakes, jungles have really become "a jungle out there".

A family of snakes has decided to cross one of these infernal roads. The road is a straight East–West band of width w, and it is a one-way road: cars travel from East to West. The family plans to leave its burrow on the South side and relocate to a nice and bushy tree right across the road on the North side, exactly on the other side of the road. The plan of the family is to cross the road perpendicularly, going from their burrow to the tree. As the snakes are diurnal, they can only travel during daylight, so their head must start to cross the road not earlier than t_1 and their tail must reach the other side of the road not later than t_2 ($t_2 > t_1$). All the snakes are moving at the same constant speed on the same line (they are so thin that they can be superimposed). Yet, due to different ages and maturity, not all of the snakes have the same length.

Cars are driving on the road at a constant speed of 1 meter per second and on a trajectory perfectly parallel to the road. Their width is exactly the same as the width of the road, and their length is so small that they can be considered as a simple line. The cars are a lethal hazard to the snakes: at the instant where a car passes over the crossing path, then the car kills any snake which is currently crossing the road.

Now, thanks to their capacity of sensing vibrations on the ground, the snake family knows the exact position, at time $t = 0$, of each car that will pass on the road today; and based on this, they can decide when they want to cross.

Your goal is to determine how many members of the family are able to make it to the other side of the road without being killed.

Example

The first test case of the sample input is illustrated in Figure 1, with cars being numbered according to their order in the input. The snake with length 1 can cross between cars 2 and 3, but the snake with length 5 has no opportunity between t_1 and t_2 to cross safely.

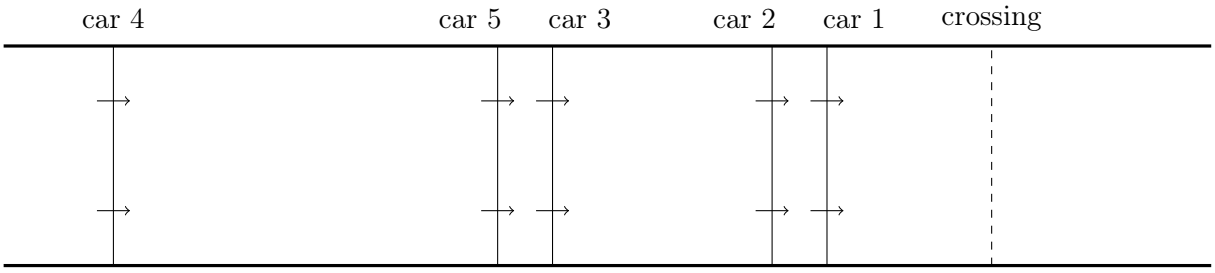


Figure 1: Roads and cars in the first test case of the sample input

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows. The first line of a test case consists of six integers s, c, w, u, t_1, and t_2, each separated by a single space:

- $1 \leq s \leq 100\,000$ is the number of snakes;
- $1 \leq c \leq 100\,000$ is the number of cars;
- $1 \leq w \leq 100\,000$ is the width of the road (in meters);
- $1 \leq u \leq 100\,000$ is the speed of the snakes (in meters per second);
- $0 \leq t_1 \leq 1\,000\,000\,000$ is the earliest possible departure time of a snake (in seconds);
- $0 \leq t_2 \leq 1\,000\,000\,000$ is the latest possible arrival time of a snake (in seconds), with $t_1 < t_2$.

The next s lines of the test case each consist of a single integer $1 \leq l_i \leq 1\,000\,000\,000$ for $1 \leq i \leq s$ indicating the length (in meters) of the i-th snake: note that multiple snakes may have the same length. Last, the next c lines of the test case each consist of a single integer $0 \leq x_j \leq 1\,000\,000\,000$ for $1 \leq j \leq c$ indicating the distance at time $t = 0$ (in meters) between the initial position of the car and the crossing path of the snakes: all cars start at the East of that crossing path, and you may assume that each car has a different starting position.

Output

For each test case in the input, your program should produce one line consisting of one integer that indicates how many of the snakes can make it to the other side. There should be no blank lines in your output.

Sample Input

```
2
2 5 4 2 3 9
1
5
3
4
8
16
9
4 2 10 5 2 5
2
5
8
10
8
3
```

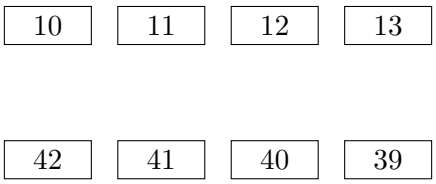
Sample Output

```
1
0
```

Problem B: Finding Your Coach

Time limit: 5 seconds

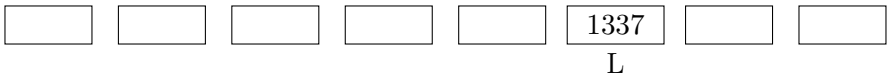
Taking a train can be such a stressful experience! You have to find the best price for your ticket, find the right station, get there on time, find the right hall and track, etc. Laurie has managed to do all of this, and the last step is to get into the right coach. Each coach has a number, but the exact manner in which coaches are numbered is a secret known only to the train company. The only rule is that coaches are numbered by consecutive positive integers, but we do not know where the numbering starts and ends, and in which direction it goes. For instance, here are two valid ways to number the coaches of a train with four coaches:



Laurie has just arrived on the platform in front of one of the coaches, and only knows the number of that coach, and how many coaches are to the left and right. Of course, Laurie needs to find one specific coach, whose number is written on the train ticket. Your job is to tell Laurie what to do next, among four possible actions:

- `G` for "Get in": get into the coach, if Laurie happens to be in front of the right coach;
- `L` for "Left": if we know that Laurie's coach is necessarily to the left;
- `R` for "Right": if we know that Laurie's coach is necessarily to the right;
- `E` for "Explore": if we do not have enough information to know in which direction Laurie's coach is, so that Laurie will have to figure it out by trial and error.

For example, consider the following situation: Laurie is in front of coach 1337, there are five coaches to the left and two coaches to the right:



If Laurie is looking for coach 1340, then necessarily it must be to the left. However, if Laurie is looking for coach 1339, then we do not have information to conclude.

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows, and consists of one single line that consists of four integers l, r, n, and m, each separated by a single space:

- $0 \leq l \leq 1\,000\,000\,000$ is the number of coaches that are strictly to the left of Laurie;
- $0 \leq r \leq 1\,000\,000\,000$ is the number of coaches that are strictly to the right of Laurie;

- $1 \leq n \leq 1\,000\,000\,000$ is the number of the coach in front of Laurie;
- $1 \leq m \leq 1\,000\,000\,000$ is the number of the coach that Laurie is looking for, i.e., the number written on Laurie's ticket.

It is guaranteed that, for each test case, there is at least one way to number the coaches of the train with consecutive positive integers which is consistent with what Laurie is seeing and with what Laurie's train ticket indicates.

Output

For each test case in the input, your program should produce one line which consists of exactly one character (followed by a newline): the character should be `G`, `L`, `R`, `E`, depending on what Laurie should do. There should be no blank lines in your output.

Sample Input

```
5
1 2 11 11
0 3 42 40
5 2 1337 1339
5 2 1337 1340
10 10 11 1
```

Sample Output

```
G
R
E
L
E
```

Problem C: Pipe Hype

Time limit: 5 seconds

As everyone knows, *Flubbyplaxmol* is this incredible slimy fluid that carries stunning energetic properties and that might fuel in a near future the voyage of a human spacecraft beyond our solar system to colonize new Goldilocks planets. Your friend from the Terrifically Powerful Transports Laboratories (or simply TPT Labs) is working on a *Flubbyendoslasher*, which is a quite simple device that routes Flubbyplaxmol between n entry gates and n exit gates. In a Flubbyendoslasher, the Flubbyplaxmol enters by different gates. It flows through pipes and finally exits by zero, one, or many gates. As Flubbyplaxmol does not mix well, no flows in a Flubbyendoslasher can be mixed. In particular, two different entry gates are never connected to the same exit gate.

Your friend is a specialist of *t-folded Flubbyendoslashers*, or t-fFs for short: they are a revolutionary multilayered stack of t identical Flubbyendoslashers, where the number of copies t can be *extremely* large. In a t-fF, the Flubbyplaxmol that exits by the i^{th} exit gate of the j^{th} Flubbyendoslasher enters immediately into the i^{th} entry gate of the $(j+1)^{\text{th}}$ Flubbyendoslasher. Note that some exit gates may not be connected to an entry gate, and vice-versa: in that case, no Flubbyplaxmol enters or exits by these gates.

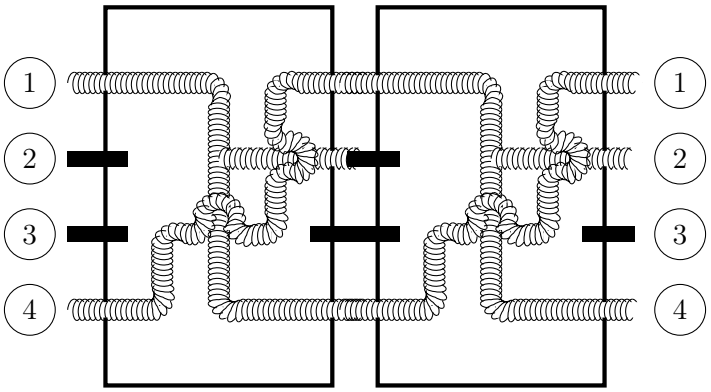


Figure 2: Illustration of a 2-folded Flubbyendoslasher

After much research, your friend has come up with a remarkable observation: *a t-fF is always equivalent to a single Flubbyendoslasher*! This is an incredible opportunity to make considerable plumbing optimizations in the race for the stars! However, to put this result to industrial uses, your friend needs your help. You are given the value of the integer t, and the description of the initial Flubbyendoslasher that is used to build the t-fF. Your goal is to compute a description of a Flubbyendoslasher that is equivalent to the t-fF.

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows. The first line of a test case consists of three integers n, m, and t, each separated by a single space:

- $1 \leq n \leq 10^6$ is the number of gates in the Flubbyendoslasher;
- $1 \leq m \leq n$ is the number of pipes between an entry gate and an exit gate in the Flubbyendoslasher;

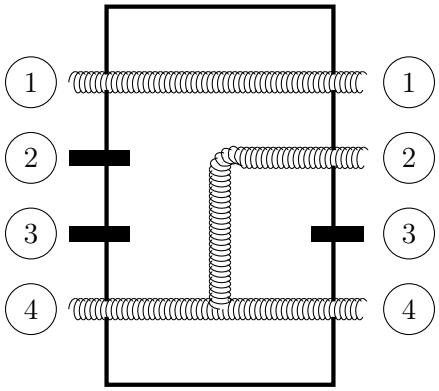


Figure 3: Flubbyendoslasher which is equivalent to the 2-fF in Figure 2

- $1 \leq t \leq 10^{15}$ is the number of copies of the Flubbyendoslasher that have been stacked in the folded Flubbyendoslasher.

The rest of the test case consists of m lines. For $1 \leq i \leq m$, the i-th line describes the i-th pipe of the Flubbyendoslasher, and consists of two integers a_i and b_i separated by a single space: it indicates that Flubbyplaxmol introduced in the entry gate $1 \leq a_i \leq n$ of the Flubbyendoslasher propagates until its exit gate $1 \leq b_i \leq n$. The pipes are given in an arbitrary order. It may be the case that the same entry gate occurs multiple times, but each exit gate occurs at most once.

Output

For each test case in the input, your program should produce a description of the Flubbyendoslasher to which the t-folded Flubbyendoslasher is equivalent. The output of your program for each test case should start with one first line consisting of one integer r that is the number of pipes. The rest of the output of your program for that test case should consist of r lines. For $1 \leq j \leq r$, the j-th line should consist of two integers x_j and y_j (with $1 \leq x_j \leq n$ and $1 \leq y_j \leq n$) separated by a single space, indicating that Flubbyplaxmol introduced in the entry gate x_j of the t-folded Flubbyendoslasher propagates until its exit gate y_j. These r lines must all be distinct and must be given in lexicographical order, i.e., they should be sorted first by the value of the entry gate, and second by the value of the exit gate. There should be no blank lines in your output.

Sample Input

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

Sample Output

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

Problem D: Twenty Thousand Leagues Under the Sea

Time limit: 5 seconds

The octoplusplus is a very smart animal that can easily tell if it can escape from any given trap. Indeed, each octoplusplus has the knowledge of its own minimal section M: an octoplusplus can pass through any hole whose area is no less than M. (The octoplusplus is very flexible, so the shape of the hole does not matter: only the area does.) Hence, when trapped in a cage, the octoplusplus can escape if and only if one of the holes in the cage has area $\geq M$.

Before his mission "Twenty Thousand Leagues Under the Sea", professor Pierre Aronnax needs to capture such an animal. He has a set of N cubic cages. The walls of these cages are grids with rectangular holes. For each cage, Pierre knows the length and width of the biggest hole. Pierre also has special glasses that determine the minimal section of any octoplusplus that he sees.

Pierre has just seen a splendid octoplusplus and wants to trap it in the cage with the biggest hole (in terms of area) while ensuring that the octoplusplus cannot escape. Can you help him?

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows. The first line of a test case consists of two integers M and N separated by a single space: $1 \leq M \leq 50\,000$ is the minimal section of the octoplusplus (in square millimeters), and $1 \leq N \leq 10\,000$ is the number of cages that Pierre owns. The rest of the test case consists of N lines. For $1 \leq i \leq N$, the i-th line describes the i-th cage, and consists of two integers L_i and W_i separated by a single space: $1 \leq L_i \leq 1\,000$ is the length (in millimeters) of the biggest hole of the i-th cage, and $1 \leq W_i \leq 1\,000$ is the width (in millimeters) of that hole. It is never the case that the largest holes of two different cages have the same area.

Output

For each test case in the input, your program should produce one line consisting of an integer $1 \leq p \leq N$ indicating the position (in the input) of the cage that satisfies Pierre's wishes, i.e., has the biggest hole in terms of area, but ensures that the octoplusplus cannot escape. If the octoplusplus is too small and no cage is adequate, you should instead produce one line consisting of the string `Too small` (followed by a newline). There should be no blank lines in your output.

Sample Input

```
2
8 5
7 7
2 3
9 9
5 8
2 4
9 2
3 5
2 5
```

Sample Output

```
2
Too small
```

Problem E: Cruise Quail

Time limit: 5 seconds

The *Cruise Quail* is a mysterious flightless bird that lives on the ground of *Madagascar's Sub-Tropical* forest (MST forest). Like salmons, which go up and down the same river at given dates, cruise quails have specific migration patterns even though they cannot fly. These patterns are the main research topic of scientists in the *Union of Animal Finders*.

Over the years, these scientists have identified a list of *nesting areas*, where cruise quails may nest, as well as *routes* connecting the nesting areas, along which cruise quails may walk. They know that every cruise quail has one nesting area for summer and another nesting area for winter (which is necessarily different). During the yearly migration, each cruise quail goes from its summer nesting area to its winter nesting area by taking a *migration path* that follows the routes (it may pass through other nesting areas along the way); and then the quail will go back from its winter nesting area to its summer nesting area following a different migration path. A cruise quail always takes a completely different path to get back to the summer nesting area, i.e., no route can be part of the two migration paths.

To learn more about cruise quails, the scientists intend to photograph them, using cameras that should be installed in the MST forest along some of the routes. When a route is equipped with a camera, then the camera will automatically trigger when a cruise quail passes by (in either direction). However, it is not always easy to equip routes with cameras, and each route thus has a *cost* that indicates how costly it is to install a camera on that route.

The scientists want to take a picture of each cruise quail every year, while minimizing the cost spent to install cameras. Specifically, they want to ensure that no matter which nesting areas and migration paths are followed by the cruise quails, it is the case that every cruise quail will pass by at least one route equipped with a camera along one of its two migration paths; and they want to know what is the minimal total cost of installing cameras to guarantee this.

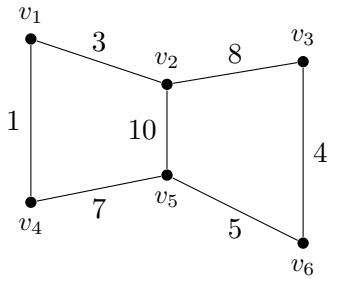


Figure 4: MST forest example

Consider, for instance, the nesting areas and routes in Figure 4, where the cost of installing a camera on a route is indicated as an integer next to it. Cruise quails can migrate along any route in any direction. A possible path for the migration of a cruise quail could be $(v_1, v_2, v_5, v_4, v_1)$. For this particular path, the cost-optimal solution would be to install a camera on the route from v_1 to v_4. Considering all possible routes that cruise quails can take, the cost-optimal solution on this example is to place cameras along routes from v_1 to v_4 and from v_3 to v_6, at a total cost of 5 (i.e., $1 + 4$). This ensures that every possible migration path passes by at least one route that is equipped with a camera.

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows. The first line of each test case consists of two integers p and r separated by a single space. The integer $3 \leq p \leq 2\,000$ is the number of nesting areas, and the integer $3 \leq r \leq 400\,000$ is the number of routes. The rest of the test case consists of r lines. For $1 \leq i \leq r$, the i-th line describes the i-th route, and consists of three integers u_i, v_i, and c_i, each separated by a single space: $1 \leq u_i \leq p$ and $1 \leq v_i \leq p$ are the nesting areas connected by the i-th route, and $1 \leq c_i \leq 3\,000$ is the cost of equipping the i-th route with a camera. There are no loops, i.e., we have $x_i \neq y_i$ for all $1 \leq i \leq r$. Furthermore, all migration routes are distinct, i.e., for all i, j with $1 \leq i < j \leq r$, we have $(u_i, v_i) \neq (u_j, v_j)$ and $(u_i, v_i) \neq (v_j, u_j)$.

Output

For each test case in the input, your program should produce one line consisting of one integer that indicates the minimum total cost of equipping routes of the MST forest with cameras so as to guarantee that each cruise quail will pass by at least one route equipped by a camera during its migration. There should be no blank lines in your output.

Sample Input

```
1
6 7
1 2 3
1 4 1
2 3 8
2 5 10
3 6 4
4 5 7
5 6 5
```

Sample Output

```
5
```

Problem F: Willy Feels Guilty

Time limit: 5 seconds

Community supported agriculture (CSA) is a great model for a food system that connects producers and consumers through a subscription process. Willy is a long-time advocate of the CSA system. After paying his membership dues, Willy knows that he will receive every now and then a new basket containing one organic product that was grown by a local farmer.

This system is great in terms of ethics, but it has one major drawback: Willy does not decide what he receives. This is a problem because Willy is also a very picky eater. Sometimes, after eating local cabbage all winter, he may prefer a juicy pineapple directly bought at the supermarket. Sometimes, after eating too many carrots, he may get fed up and throw away some of them. Sometimes, he may decide to exchange some of his vegetables with friends. In fact, Willy has already decided on his exact menu, and will only eat this specific sequence of products (in order). Nevertheless, because Willy firmly believes in the CSA model, he feels guilty every time he deviates from his subscription.

Your job is to help Willy manage the incoming deliveries to match exactly his tastes while minimizing the guilt cost. You are given the list of products that Willy will receive from his CSA plan (in that order), and Willy's menu of the products that he will eat (in that order). Willy can deviate from the subscription in three ways: throwing away a product, buying a product from the supermarket, or exchanging one product for another. You are also given the guilt cost of each of these actions (which is incurred each time they are performed). What is the minimal guilt cost that Willy needs to pay so that he gets to eat exactly according to his menu?

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows, and consists of three lines. The first line of a test case consists of three integers b, t, and s, each separated by a single space:

- $0 \leq b \leq 1000$ is the guilt price of buying a product;
- $0 \leq t \leq 1000$ is the guilt price of throwing away a product;
- $0 \leq s \leq 1000$ is the guilt price of swapping one product, i.e., Willy exchanges one single product occurrence in his subscription for some other arbitrary product obtained from a friend.

The second line of the test case indicates the list of the products that Willy will receive from his CSA subscription: it starts with an integer $1 \leq n \leq 1000$ indicating the number of products, which is followed by a single space and a list of n product names separated by single spaces. Each product name is a nonempty string of at most 20 lowercase letters between 'a' and 'z'. The third line of the test case indicates Willy's menu, and is a list of products formatted like the second line. The same product name may occur multiple times, in the same line or in multiple lines. It is possible for a product name to occur only on the second line, or only on the third line.

Output

For each test case in the input, your program should produce one line consisting of one integer that represents the least guilt cost that Willy needs to experience in order to eat exactly the sequence of products in his menu. There should be no blank lines in your output.

Sample Input

```
4
3 5 0
3 turnip cabbage apple
2 tomato apple
1 1 10
5 carrot carrot carrot carrot carrot
3 carrot carrot apple
3 6 5
2 apple carrot
2 carrot apple
1 1 100
4 cabbage cabbage cabbage pineapple
4 pineapple apple apple apple
```

Sample Output

```
5
4
9
6
```

Problem G: The Total is Right

Time limit: 5 seconds

The total is right is a popular TV game that has been broadcast in many countries. The goal of the game is to reach a chosen number N using the four arithmetic operations ($+$, $-$, $\times$, $\div$) and six given integer numbers m_i for $1 \leq i \leq 6$. All the intermediate numbers must be positive and any division must be exact (for example, it is not possible to divide 5 by 2). The input numbers m_i as well as all intermediate numbers can be used at most once, but you are not required to use all input numbers.

For instance, suppose $N = 888$ and $m_1 = 100$, $m_2 = 6$, $m_3 = 75$, $m_4 = 3$, $m_5 = 1$, and $m_6 = 6$. We can then obtain $N = 888$ by computing

$$75 - 1 = 74 \qquad \text{(using } m_3/m_5\text{)}$$
$$6 + 6 = 12 \qquad \text{(using } m_2/m_6\text{)}$$

and finally

$$74 \times 12 = 888. \qquad \text{(using } m_3/m_5 \text{ and } m_2/m_6 \text{ respectively)}$$

Hence, in that example, the total is right!

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows, and consists of one single line that consists of seven integers N, m_1, m_2, m_3, m_4, m_5, and m_6, each separated by a single space. The integer $101 \leq N \leq 999$ is the number that must be reached, and the integers m_i are the numbers that can be used to reach N: for each $1 \leq i \leq 6$, we have $m_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100\}$.

Output

For each test case in the input, your program should print one line consisting of the string `The total is right` (followed by a newline) or of the string `Impossible` (followed by a newline), depending on whether or not it is possible to obtain exactly N using some or all of the numbers m_i for $1 \leq i \leq 6$ according to the rules of the game. There should not be any blank lines in your output.

Sample Input

```
2
888 100 6 75 3 1 6
449 2 6 100 10 2 8
```

Sample Output

```
The total is right
Impossible
```