# Problem 1 – Key Revolver

Our favorite super-spy action hero Sam is back from his mission in another exam, and this time he has an even more difficult task. He needs to **unlock a** **safe**. The problem is that the safe is **locked** by **several locks in a row**, which all have **varying** **sizes**.

Our hero posesses a special weapon though, called the **Key Revolver**, with special bullets. Each **bullet** can unlock a **lock** with a **size** **equal to or larger** **than** the **size** of the **bullet**. The bullet goes into the keyhole, then explodes, completely **destroying** it. Sam **doesn’t know the size** of the locks, so he needs to just shoot at all of them, until the safe runs out of locks.

What’s behind the safe, you ask? Well, intelligence! It is told that Sam’s sworn enemy – **Nikoladze**, keeps his **top secret** **Georgian** **Chacha Brandy** recipe inside. It’s valued differently across different times of the year, so Sam’s boss will tell him what it’s worth over the radio. One last thing, every bullet Sam fires will also cost him money, w**hich will be deducted from his pay** from the price of the intelligence.

Good luck, operative.

## Input

* On the **first line** of input, you will receive the price of each **bullet** – an **integer** **in the range [0-100]**
* On the **second line**, you will receive the **size of the gun barrel** – an **integer** **in the range [1-5000]**
* On the **third line**, you will receive the **bullets** – a **space-separated integer sequence** with **[1-100] integers**
* On the **fourth line**, you will receive the **locks** – a **space-separated integer sequence** with **[1-100] integers**
* On the **fifth** **line**, you will receive the **value of the intelligence** – an **integer** **in the range [1-100000]**

After Sam receives all of his information and gear (**input**), he starts to **shoot the locks** **front-to-back**, while going through the bullets **back-to-front**.

If the **bullet** has a **smaller or equal** size to the **current** **lock**, print “Bang!”, then **remove the lock**. If not, print “Ping!”, leaving the lock **intact**. The bullet is removed in **both cases**.

If Sam runs out of bullets in his barrel, print “Reloading!” on the console, then continue shooting. If there aren’t any bullets left, **don’t** print it.

The program ends when Sam **either** **runs out of bullets**, or the safe **runs out of** **locks**.

## Output

* If Sam **runs out of bullets** before the safe runs out of **locks**, print:  
  “Couldn't get through. Locks left: {locksLeft}”
* If Sam manages to **open the safe**, print:  
  “{bulletsLeft} bullets left. Earned ${moneyEarned}”

Make sure to account for the **price of the bullets** when calculating the **money earned**.

## Constraints

* The input will be **within the constaints** specified above and will **always be valid**. There is **no need** to check it explicitly.
* There will **never** be a case where Sam breaks the lock and ends up with **negative balance**.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 50  2  11 10 5 11 10 20  15 13 16  1500 | Ping!  Bang!  Reloading!  Bang!  Bang!  Reloading!  2 bullets left. Earned $1300 | 20 shoots lock 15 (ping)  10 shoots lock 15 (bang)  11 shoots lock 13 (bang)  5 shoots lock 16 (bang)  Bullet cost: 4 \* 50 = $200  Earned: 1500 – 200 = $1300 |
| 20  6  14 13 12 11 10 5  13 3 11 10  800 | Bang!  Ping!  Ping!  Ping!  Ping!  Ping!  Couldn't get through. Locks left: 3 | 5 shoots lock 13 (bang)  10 shoots lock 3 (ping)  11 shoots lock 3 (ping)  12 shoots lock 3 (ping)  13 shoots lock 3 (ping)  14 shoots lock 3 (ping) |
| 33  1  12 11 10  10 20 30  100 | Bang!  Reloading!  Bang!  Reloading!  Bang!  0 bullets left. Earned $1 | 10 shoots lock 10 (bang)  11 shoots lock 20 (bang)  12 shoots lock 30 (bang)  Bullet cost: 3 \* 33 = $99  Earned: 100 – 99 = $1 |

# Problem 2 – Sneaking

After our hero Sam got the recipe from the first problem, there is another thing he needs to check off from his to-do list. In order to make the recipe even more valuable, he needs to “eliminate” anyone who possesses the knowledge of it. That person is Sam’s sworn enemy - **Nikoladze**. Sam needs to get through a rectangular room of **patrolling** **enemies** until he finally **reaches Nikoladze**.

A standard room looks like this:

|  |  |
| --- | --- |
| **Room** | **Legend** |
| ......N... b......... ..d....... ......d... .....S.... | S 🡺 **Sam**, the player character  b/d 🡺 **left/right-facing** **patrolling enemy**  N 🡺 **Nikoladze**  . 🡺 **Empty space** |

Each turn proceeds as follows:

* **First, Enemies** move either **left** or **right**, depending on which **direction** they are **facing** (b goes **right**, d goes **left**)
  + If an enemy is standing on the **edge** of the room, he flips his **direction** (from d to b or from b to d) and **doesn’t move** for the rest of the turn.
* If an enemy is on the **same row** as Sam, and also **facing** **Sam** (eg. .b.S.), the **enemy** **kills Sam**.
* After that, Sam moves in the **direction** he is instructed to (either U/D/L/R or W).
  + **U** -> **Up**, **D** -> **Down**, **L** -> **Left**, **R** -> **Right**, **W** -> **Wait (Sam doesn’t move)**
* If **Sam** moves **onto an enemy** (**same row** and **column**), Sam **kills** the enemy and **leaves no trace of him**.
* If Sam is reaches the **same row** as **Nikoladze**, **Sam** kills **Nikoladze** (replacing him with an **X**)

## Input

* On the **first line** of input, you will receive n – the **number of rows** the **room** will consist of. Range: **[2-20]**
* On the next **n lines**, you will receive the **room**, which Sam will have to navigate.
* On the **final line** of input, you will receive a sequence of **directions** – one of (**U**, **D**, **L**, **R**, **W**)

## Output

* If Sam is **killed**, print “Sam died at {row}, {col}”
* If Nikoladze is **killed**, print “Nikoladze killed!”
* Then, in both cases, **print** the **final state of the room** on the **console**, with either **Sam** or **Nikoladze’s** **symbols** replaced by an X.

## Constraints

* The room will always be **rectangular**.
* There will **always** be enough moves for **Sam** to reach **Nikoladze**
* There will be **no case** where **Sam** is instructed to move **out of the bounds of the room**.
* There will be **no case** with **two enemies on the same row**.
* There will be **no case** with an **enemy and Nikoladze** standing on the **same row**.
* There will be **no case** where Sam reaches the **same** **row and column** as **Nikoladze**.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5  ......N...  b.........  ..d.......  ......d...  .....S....  UUUUR | Sam died at 2, 5  ......N...  ...b......  b....X....  ..........  .......... | Turn 1: Enemies move, then Sam steps on the enemy on the 4th row.  Turn 2: Enemies move, then Sam moves.  Turn 3: Enemy 2 turns around, sees Sam and kills him. |
| 3  N......  .b.....  ..dS...  WUUU | Nikoladze killed!  X..S...  .......  b...... | Turn 1: Enemies move, Sam waits.  Turn 2: Enemies move, Sam goes up, steps on an enemy.  Turn 3: Enemies move, Sam goes up, kills Nikoladze. |
| 6  .............  ....S........  .b...........  ...........d.  .............  ....N........  WWWDWWWDDRD | Nikoladze killed!  .............  .............  ............b  d............  .............  ....XS....... | Turn 1/2/3: Enemies move, Sam waits.  Turn 4: Enemies move, Sam goes down.  Turn 5/6/7: Enemies move, Sam waits.  Turn 8/9: Enemies move, Sam goes down.  Turn 10: Enemies move, Sam goes right.  Turn 11: Enemies move, Sam goes down and kills Nikoladze. |

# Problem 3 – Crypto Blockchain

The next task for our hero Sam is to **hack the main top-secret facility server**, used to manage all of Nikoladze’s social media. He’s already reached the server, and now it’s time to decrypt the information on it to see if it’s valuable or not. Luckily, you’re Sam’s top unpaid intern, and he has tasked you with figuring out the algorithm to decrypting the data. So, plug in some headphones and put on some hacker music. It’s time to **decrypt the** **Crypto Blockchain**.

The **Crypto Blockchain** is a special **sequence of characters**, which is comprised of **several lines**. Each line is **always 16 characters long**. Inside these lines, there are several **Crypto Blocks** and some garbage data around them. Here’s what a sample **Crypto Blockchain** looks like:

|  |
| --- |
| OktJULP\{FT\*n\*uk  \_123120137130v}M  OoHw\_[1291201341  34r`wkR]00000000 |

The first step is to **condense** the **Crypto Blockchain** into **one line**.

The next step is to search for **special substrings** inside it, called **Crypto Blocks**. Each valid **Crypto Block** has the following characteristics:

* It’s **enclosed** in either **brackets** {} or **square brackets** [].
  + If it contains **mixed opening/closing brackets** (such as {] or [}, **ignore that Crypto Block entirely**)
* It contains **any printable ASCII character** inside it
* It contains **at least three** **digits** **in a row**.
  + If the number of digits it contains **cannot be split into threes** (e.g. 8 digits), **ignore the Crypto Block**.

We’re looking for the **digits** inside each **Crypto Block**, which are actually **encoded** **ASCII characters**. Each character is represented by **3 digits** (**converted to a number**), and the **sequence of digits** can be split into threes to figure out the sequence of characters present in that crypto block.

Looking at these characteristics, we can look at the above expanded Crypto Blockchain and **find all the Crypto Blocks** (**green** represents the **entire block**, **yellow** represents the **digits** we’re looking for):

|  |
| --- |
| OktJULP\{FT\*n\*uk\_123120137130v}MOoHw\_[129120134134r`wkR]00000000 |

Once we **find** the **digits** in one crypto block, we split them into **threes** and **convert them** to a **string of characters** by **subtracting the length** of the **entire crypto block** from **each number individually**.

The final step is performing this algorithm over **all the crypto blocks individually** and **concatenating** the result.

## Input

* On the **first line** of input, you will receive **n** – the **number of rows** the **room** will consist of
* On the next **n lines**, you will receive the **Crypto Blockchain**, a sequence of **16** characters.

## Output

* Print the **decrypted** and **concatenated** text.

## Constraints

* Crypto blocks will always contain **zero or one sequence of numbers**.
* There will **always** be a **valid** **crypto block** in each crypto blockchain.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 4  OktJULP\{FT\*n\*uk  \_123120137130v}M  OoHw\_[1291201341  34r`wkR]00000000 | darkness | Block 1: {FT\*n\*uk\_123120137130v}  Numbers: 123, 120, 137, 130. Crypto Block Length: 23  Subtracted ASCII codes: 100, 97, 114, 107 🡺 dark  Block 2: [129120134134r`wkR]  Numbers: 129, 120, 134, 134. Crypto Block Length: 19  Subtracted ASCII codes: 110, 101, 115, 115 🡺 ness |
| 7  [>K.l ~T11715215  2153081069148155  138z]#YQej@<+;|[  1370551271241371  24056]aG\'#|J q{  L|y!111632]!u<@:  <-&D000000000000 | Psst, over here! | Block 1: [>K.l ~T117152152153081069148155138z]  Numbers: 117, 152, 152, 153, … Crypto Block Length: 37  Subtracted ASCII codes: 80, 115, 115, 116, … 🡺 Psst, ove  Block 2: [137055127124137124056]  Numbers: 137, 55, 127, 124, … Crypto Block Length: 23  Subtracted ASCII codes: 114, 32, 104, 101, … 🡺 r here!  Block 3: {L|y!111632]  Brackets are different 🡺 ignore |
| 4  [099134134130055  142127]{12614506  1091102089061131  140}[128121111]0 | Look what >I< found | Block 1: [099134134130055142127]  Numbers: 99, 134, 134, 130, … Crypto Block Length: 23  Subtracted ASCII codes: 76, 111, 111, 107, … 🡺 Look wh  Block 2: {126145061091102089061131140}  Numbers: 126, 145, 61, 91, … Crypto Block Length: 29  Subtracted ASCII codes: 97, 116, 32, 62, … 🡺 at >I< fo  Block 3: [128121111]  Numbers: 128, 121, 111. Crypto Block Length: 11  Subtracted ASCII codes: 117, 110, 100, … 🡺 und |

# Problem 4 – Hit List

One final task for Sam before he gets to go home... Data mining!

Sam will receive info about one or several people in the format

* “{name}={key}:{value};{key}:{value};…”.

The goal here is to **group** the info for every person by their **name**. If a **key** is received **multiple times**, keep **only the most recent value**.

On the last line, you will receive “Kill {name}”. Your task is to find all the info on that name and print it, **ordered alphabetically by key**.

Then, Sam needs to build a so-called **info index** on them. The **info index** is comprised of the **sum** of all the **keys’ lengths** and **values’ lengths** of that person’s info.

If the **info index** is **larger or equal to** the **target info index** (given on the **first line** of input), print “Proceed”. Otherwise, print “Need {infoNeeded} more info.“.

## Input

* On the **first line**, you will receive the **target info index**, an **integer** in the **range** **[25-90]**
* Until you receive the text “end transmissions”, **keep reading** **new lines** with information.
* On the final line, you will receive “Kill {name}”

## Output

* On the first line, print “Info on {name}:”.
* On the next lines, print “---{info}: {value}”
* On the next line, print “Info index: {infoIndex}” with the **info index** of the selected person.
* On the final line, print either “Proceed” or “Need {infoNeeded} more info.”, depending on whether the info is **enough** to carry out the hit or not.

## Constraints

* There will **always** be at least **one** name in the input.
* Each name will **always** have **one or several** **key/value** pairs associated with it.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 30  Kobin=age:20;salary:700  Grimsdottir=salary:5000  Kobin=education:High School  end transmissions  Kill Kobin | Info on Kobin:  ---age: 20  ---education: High School  ---salary: 700  Info index: 34  Proceed |
| 20  Lambert=age:57;salary:7000  Grimsdottir=salary:5000  John=salary:1550  John=lastName:Smith  John=salary:1800  Kobin=education:High School  end transmissions  Kill Lambert | Info on Lambert:  ---age: 57  ---salary: 7000  Info index: 15  Need 5 more info. |
| 25  Bill=salary:900;lastName:Billov  Kobin=salary:1300  Kobin=education:High School  end transmissions  Kill Bill | Info on Bill:  ---lastName: Billov  ---salary: 900  Info index: 23  Need 2 more info. |