# Problem 2 – SpyGram

After arriving from the trip from the Splinter Trip problem, our hero **Sam** is feeling a bit jet lagged, but he’s ready to go to work! He needs to receive **orders** from his **commanding officers** through the sophisticated messaging app, called **SpyGram**. The app uses a sophisticated algorithm to **encrypt** messages. Since you’re tired of jumbling text by hand for 8 hours a day, you decide to write an **algorithm** to do it for you.

**Sending** messages works the following way – an **outgoing** message **must** follow the following **format**:

* TO: {recipient}; MESSAGE: {message};

Here’s what an example message looks like: “TO: GOSHO; MESSAGE: hi.;”. If any message is **not** in this format, you should **ignore it**.

Both parties are given a **private key**, which consists of **digits** of a **variable length**, with which to **encrypt**/**decrypt** messages. The encryption algorithm is simple:

First, we take the message in the format above, then we **shift right** the **first character** of our message by the **value** of the **first character** of our **private key** in the **ASCII table,** the **second character** by the **second private key character,** and so on. If we **run out** **of characters** in our **private key** (such as when our message is longer than our private key), then we **start over** from the beginning of our **private key**.

So, with an **example message** of “**hello”**, and an **example private key** of **123**, it would look like this:

hello -> igomq

* h gets shifted right by **1 character**
* e gets shifted right by **2 characters**
* l gets shifted right by **3 characters**
* l gets shifted right by **1 character** (we **ran out** of characters in the **private key** and **rolled over** to the start)
* o gets shifted right by **2 characters**.

### Input

The **first line** of our input consists of our **private key**.

Until we receive the command “**END**”, we’ll start receiving **non-encrypted messages to send** to the command center. If we receive a **message to** **send**, we need to put it in our **pending messages** collection.

After that, we need to **sort** the sent messages by **sender name** in **ascending order**.

### Output

To send a message, all we have to do is **encrypt it** and **print** it on the console in the standard **outgoing message** format.

### Constraints

* All **valid** recipient/sender names will be **UPPERCASE** and contain only **Latin letters**
* Messages can contain **ANY** **ASCII character**Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 13234  TO: GRIM; MESSAGE: hello;  TO: ARCHER; MESSAGE: sneak around it;  END | UR<#ESFJHV<#OHWTDIH>!vphel#cusvqf#mu>  UR<#KSLO>$NHUVEHH<#lfonr? |
| 142325555  TO: KOBIN; MESSAGE: one two three;  TO: KESTREL; MESSAGE: affirmative;  T: REGAN; MESSAGE: help me;  TO: TOMCLANCY; MESSAGE: let's get to work;  TO: kestrel; MESSAGE: affirmative;  END | US<#MJXYWFP=#OJXXFHI<#ckknwnevlxj@  US<#MTGNS<$OHUXFLJ;$qqg%y|t!xjugj@  US<#VTRHQBRE\=%RJXTEIH<%qjy(w"jgy%yt!{qum@ |
| 82738  TO: ARCHER; MESSAGE: affirmative;  FROM: SAM; MESSAGE: i'm pinned down;  TO: SAM; MESSAGE: 55% done;  FROM: SAM; MESSAGE: infiltrate the storage facility;  END | \QA#IZEOHZC"TH[[CNHB(cmiqzohwq~gB  \QA#[IOB#UMUZDOM<'8=-"krvm= |

Докато сме на тема Типове Данни, може ли да се отделят няколко минути да тип „ dynamic“ . Имаше задача, която чете числа от списък и се извършват прости математически операции с тях. В случай, когато входа не съдържа числа и списъка остане празен,една променлива от „инт“ трябва да приеме стойност „None“ (string). Ясно е че може с проверка, но реших да я спестя с тип dynamic и в крайна сметка нищо не се получи. Защо?