# Exam Preparation

Part I: [judge.softuni.bg/Contests/960](https://judge.softuni.bg/Contests/Practice/Index/960" \l "0)

Part II: [judge.softuni.bg/Contests/962](https://judge.softuni.bg/Contests/Practice/Index/962#0)

# Problem 1. Padawan Equipment

Yoda is starting his newly created Jedi academy. So, he asked master Ivan Cho to **buy** the **needed equipment**. The number of **items** depends on **how many students will sign up**. The equipment for the Padawan contains **lightsabers, belts and robes**.

You will be given **the amount of money Ivan Cho has**, the **number of students** and the **prices of each item**. You have to help Ivan Cho **calculate** if the **money** he has is **enough to buy all of the equipment**, or how much more money he needs.   
Because the lightsabres sometimes brake, Ivan Cho should **buy 10% more**, **rounded up** to the next integer. Also, every **sixth belt is free**.

## Input / Constraints

The input data should be read from the console. It will consist of **exactly 5 lines**:

* The **amount of money** Ivan Cho has – **floating-point number** in **range [0.00…1,000.00]**
* The **count of students – integer in range [0…100]**
* The **price of lightsabers** for a **single sabre – floating-point number** in **range [0.00…100.00]**
* The **price of robes** for a **single robe – floating-point number** in **range [0.00…100.00]**
* The **price of belts** for a **single** **belt – floating-point number** in **range [0.00…100.00]**

The **input data will always be valid**. **There is no need to check it explicitly**.

## Output

The output should be printed on the console.

* **If the calculated price of the equipment is less or equal to the money Ivan Cho has:**
  + "The money is enough - it would cost {the cost of the equipment}lv."
* **If the calculated price of the equipment is more than the money Ivan Cho has:**
  + "Ivan Cho will need {neededMoney}lv more."
* **All prices** must be **rounded to two digits after the decimal point.**

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 100  2  1.0  2.0  3.0 | The money is enough - it would cost 13.00lv. | Needed equipment for 2 padawans :  sabresPrice\*(studentsCount + 10%) + robesPrice \* (studentsCount) + beltsPrice\*(studentsCount-freeBelts)  1\*(3) + 2\*(2) + 3\*(2) = 13.00  13.00 <= 100 – the money will be enough. |
| **Input** | **Output** | **Comments** |
| 100  42  12.0  4.0  3.0 | Ivan Cho will need 737.00lv more. | Needed equipment for 42 padawans:  12\*47 + 4\*42 + 3\*35 = 837.00  837 > 100 – need 737.00 lv. more. |

*...May the force  
 be with you...*

# Problem 2. Kamino Factory

The clone factory in Kamino got another order to clone troops. But this time you are tasked to find **the best DNA** sequence to use in the production.

You will receive the **DNA length** and until you receive the command **"Clone them!"** you will be receiving a **DNA sequences of ones and zeroes, split by "!"** **(one or several).**

You should select the sequence with the **longest subsequence of ones**. If there are several sequences with **same length of** **subsequence of ones**, print the one with the **leftmost** **starting index**, if there are several sequences with same **length and starting index**, select the sequence with the **greater sum** of its elements.

After you receive the last command "Clone them!" you should print the collected information in the following format:

"Best DNA sample {bestSequenceIndex} with sum: {bestSequenceSum}."

"{DNA sequence, joined by space}"

## Input / Constraints

* The **first line** holds the **length** of the **sequences** – **integer in range [1…100];**
* On the next lines until you receive **"Clone them!"** you will be receiving sequences (at least one) of ones and zeroes, **split by "!"** (one or several).

## Output

The output should be printed on the console and consists of two lines in the following format:

"Best DNA sample {bestSequenceIndex} with sum: {bestSequenceSum}."

"{DNA sequence, joined by space}"

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5  1!0!**1!1**!0  0!**1!1**!0!0  Clone them! | Best DNA sample 2 with sum: 2.  0 1 1 0 0 | We receive 2 sequences with **same length** **of subsequence of ones**, but the second is printed, because its subsequence starts at **index[1].** |
| **Input** | **Output** | **Comments** |
| 4  **1!1**!0!**1**  1!0!0!1  **1!1**!0!0  Clone them! | Best DNA sample 1 with sum: 3.  1 1 0 1 | We receive 3 sequences. Both 1 and 3 **have same length** of subsequence of ones -> 2, **and both start from index[0]**, but the first is printed, because its **sum is greater.** |

"FEAR IS THE PATH TO THE DARK SIDE...

FEAR LEADS TO ANGER...

ANGER LEADS TO HATE...

HATE LEADS TO SUFFERING...."

# Problem 3. Star Enigma

The war is in its peak, but you, young Padawan, can turn the tides with your programming skills. You are tasked to create a program to **decrypt** the messages of The Order and prevent the death of hundreds of lives.

You will receive several messages, which are **encrypted** using the legendary star enigma. You should **decrypt the messages**, following these rules:

To properly decrypt a message, you should **count all the letters** **[s, t, a, r]** – **case insensitive** and **remove** the count from the **current ASCII value of each symbol** of the encrypted message.

After decryption:

Each message should have a **planet name, population, attack type ('A', as attack or 'D', as destruction) and soldier count.**

The planet name **starts after** **'@'** and contains **only letters from the Latin alphabet**.

The planet population **starts after ':'** and is an **Integer**;

The attack type may be **"A"(attack) or "D"(destruction)** and must be **surrounded by "!"** (exclamation mark).

The **soldier count** starts after **"->"** and should be an Integer.

The order in the message should be: **planet name -> planet population -> attack type -> soldier count.** Each part can be separated from the others by **any character except: '@', '-', '!', ':' and '>'.**

## Input / Constraints

* The **first line** **holds n** – the number of **messages**– **integer in range [1…100];**
* On the next **n** lines, you will be receiving encrypted messages.

## Output

After decrypting all messages, you should print the decrypted information in the following format:

First print the attacked planets, then the destroyed planets.  
"Attacked planets: {attackedPlanetsCount}"  
"-> {planetName}"  
"Destroyed planets: {destroyedPlanetsCount}"  
"-> {planetName}"

The planets should be **ordered by name** **alphabetically.**

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  **ST**CDoghudd4=63333$D$0**A**53333  EHf**s**y**ts**nhf?8555&I&2C9555**SR** | Attacked planets: 1  -> Alderaa  Destroyed planets: 1  -> Cantonica | We receive two messages, to decrypt them we calculate the key:  First message has decryption key 3. So we substract from each characters code 3.  **PQ@Alderaa1:30000!A!->20000**  The second message has key 5.  **@Cantonica:3000!D!->4000NM**  **Both messages are valid** and they contain planet, population, attack type and soldiers count.  After decrypting all messages we print each planet according the format given. |
| **Input** | **Output** | **Comments** |
| 3  **tt**(''DG**s**vywge**r**x>6444444444%H%1B9444  GQh**rr**|**A**977777(H(**TTTT**  EHf**s**y**ts**nhf?8555&I&2C9555**SR** | Attacked planets: 0  Destroyed planets: 2  -> Cantonica  -> Coruscant | We receive three messages.  Message one is decrypted with key 4:  **pp$##@Coruscant:2000000000!D!->5000**  Message two is decrypted with key 7:  **@Jakku:200000!A!MMMM**  This is **invalid message**, missing soldier count, so we continue.  The third message has key 5.  **@Cantonica:3000!D!->4000NM** |

"It’s a trap!" – Admiral Ackbar

# Problem 4. ForceBook

The force users are struggling to remember which side are the different forceUsers from, because they switch them too often. So you are tasked to create a web application to manage their profiles. You should store information for every **unique forceUser**, registered in the application.

You will receive **several input lines** in one of the following formats:

{forceSide} | {forceUser}

{forceUser} -> {forceSide}

The forceUser and forceSide are strings, containing any character.

If you receive forceSide | forceUser you should **check if such forceUser already exists**, and **if not**, **add** him/her to the corresponding side.

If you receive a forceUser -> forceSide you should check if there is such **forceUser** already and if so, **change his/her side**. If there is no such **forceUser**, add him/her to the corresponding forceSide, treating the command **as new registered forceUser.**  
Then you should print on the console: "{forceUser} joins the {forceSide} side!"

You should end your program when you receive the command "Lumpawaroo". At that point you should print each force side, **ordered descending by forceUsers count, than ordered by name**. For each side print the **forceUsers**, **ordered by name**.

In case there are **no forceUsers in a side**, you **shouldn`t print** the side information.

## Input / Constraints

* The input comes in the form of commands in one of the formats specified above.
* The input ends when you receive the command "Lumpawaroo".

## Output

* As output for each forceSide, **ordered descending by forceUsers count**, **then by name**, you must print all the forceUsers, **ordered by name alphabetically**.
* The output format is:

Side: {forceSide}, Members: {forceUsers.Count}

! {forceUser}

! {forceUser}

! {forceUser}

* In case there are **NO** forceUsers, don`t print this side.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Light | Gosho  Dark | Pesho  Lumpawaroo | Side: Dark, Members: 1  ! Pesho  Side: Light, Members: 1  ! Gosho | We register Gosho in the Light side and Pesho in the Dark side. After receiving "Lumpawaroo" we print both sides, ordered by membersCount and then by name. |
| **Input** | **Output** | **Comments** |
| Lighter | Royal  Darker | DCay  Ivan Ivanov -> Lighter  DCay -> Lighter  Lumpawaroo | Ivan Ivanov joins the Lighter side!  DCay joins the Lighter side!  Side: Lighter, Members: 3  ! DCay  ! Ivan Ivanov  ! Royal | Although Ivan Ivanov doesn`t have profile, we **register** him and add him to the Lighter side.  We **remove DCay** from Darker side and add him to Lighter side.  We print only Lighter side because Darker side **has no members.** |

"I find your lack of faith disturbing." — Darth Vader