# Problem 1 - Black Flag

Problem for exam preparation for the [Programming Fundamentals Course @SoftUni](https://softuni.bg/courses/programming-fundamentals-csharp-java-js-python).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/Practice/Index/1773#0>.

*Pirates are invading the sea, and you're tasked to help them plunder*

Create a program that checks if **target plunder** is **reached**. First, you will receive how many **days** the pirating lasts. Then you will receive how much the pirates **plunder for a day**. Last you will receive the **expected plunder** at the end.

Calculate how much **plunder** the pirates manage to **gather**. Each **day** they gather the **plunder**. Keep in mind that they attack more ships every third day and add additional plunder to their total gain, which is **50% of the daily plunder**. Every **fifth day** the pirates encounter a warship, and after the battle, they **lose 30%** of their **total plunder**.

If the gained plunder is **more or equal** to the target, print the following:

**"Ahoy! {totalPlunder} plunder gained."**

If the gained plunder is **less** than the target. Calculate the **percentage left** and print the following:

**"Collected only {percentage}% of the plunder."**

Both numbers should be **formatted** to the **2nd decimal place**.

## Input

* On the **1st line,** you will receive the **days** of the plunder – an **integer number** in the range [0…100000]
* On the **2nd line,** you will receive the **daily plunder** – an **integer number** in the range [0…50]
* On the **3rd line,** you will receive the **expected plunder** – a **real number** in the range [0.0…10000.0]

## Output

* In the end, print whether the plunder **was successful** or **not,** following the format **described above**.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 40 100 | Ahoy! 154.00 plunder gained. |
| **Comments** | |
| The days are 5, and the daily plunder is 40. On the third day, the total plunder is 120, and since it is a third day, they gain an additional 50% from the daily plunder, which adds up to 140. On the fifth day, the plunder is 220, but they battle with a warship and lose 30% of the collected cargo, and the total becomes 154. That is more than expected. | |
|  | |
| 10  20  380 | Collected only 36.29% of the plunder. |

## JS Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| (["5", "40", "100"]) | Ahoy! 154.00 plunder gained. |
| **Comments** | |
| The days are 5, and the daily plunder is 40. On the third day, the total plunder is 120, and since it is a third day, they gain an additional 50% from the daily plunder, which adds up to 140. On the fifth day, the plunder is 220, but they battle with a warship and lose 30% of the collected cargo, and the total becomes 154. That is more than expected. | |
|  | |
| (["10",  "20",  "380"]) | Collected only 36.29% of the plunder. |

# Problem 2 - Treasure Hunt

Problem for exam preparation for the [Programming Fundamentals Course @SoftUni](https://softuni.bg/courses/programming-fundamentals-csharp-java-js-python).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/Practice/Index/1773#1>.

*The pirates need to carry a treasure chest safely back to the ship, looting along the way.*

Create a program that **manages** the **state** of the **treasure chest** along the way. On the **first line,** you will receive the **initial loot** of the treasure chest, which is a **string** of **items** separated by a **"|"**.

**"{loot1}|{loot2}|{loot3} … {lootn}"**

The following lines represent commands **until** **"Yohoho!"** which ends the treasure hunt:

* **"Loot {item1} {item2}…{itemn}"**:
  + Pick up treasure loot along the way. Insert the items at the **beginning** of the chest.
  + If an item is **already** contained, **don't** insert it.
* **"Drop {index}"**:
  + **Remove** the loot at the given **position** and **add** it at the **end** of the treasure chest.
  + If the index is **invalid,** skip the command.
* **"Steal {count}"**:
  + Someone steals the **last count** loot items. If there are **fewer items** than the given count, **remove as much** as there are.
  + Print the stolen items separated by **", "**:

**"{item1}, {item2}, {item3} … {itemn}"**

In the end, output the **average treasure gain,** which is the **sum** of all treasure items **length** divided by the **count** of all items inside the chest **formatted** to the **second decimal** point:

**"Average treasure gain: {averageGain} pirate credits."**

If the chest is **empty,** print the following message:

**"Failed treasure hunt."**

## Input

* On the **1st line,** you are going to receive the **initial treasure chest** (**loot** separated by **"|"**)
* On the following **lines**, until **"Yohoho!"**, you will be receiving commands.

## Output

* Print the output in the **format** **described** **above**.

## Constraints

* The **loot items** will be strings containing any ASCII code.
* The **indexes** will be integers in the range [**-200**…**200**]
* The **count** will be an integer in the range [**1**….**100**]

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Gold|Silver|Bronze|Medallion|Cup  Loot Wood Gold Coins  Loot Silver Pistol  Drop 3  Steal 3  Yohoho! | Medallion, Cup, Gold  Average treasure gain: 5.40 pirate credits. |
| **Comments** | |
| The first command **"Loot Wood Gold Coins"** adds **Wood** and **Coins** to the chest but **omits** Gold since it is already contained. The chest now has the following items:  **Coins Wood Gold Silver Bronze Medallion Cup**  The **second** command adds **only Pistol** to the chest  The **third** command **"Drop 3"** removes the **Gold** from the chest, but immediately adds it at the **end**:  **Pistol Coins Wood Silver Bronze Medallion Cup Gold**  The **fourth** command **"Steal 3"** removes the **last 3** items **Medallion**, **Cup**, **Gold** from the chest and prints them.  In the end calculate the average treasure gain which is the sum of all items length Pistol(**6**) + Coins(**5**) + Wood(**4**) + Silver(**6**) + Bronze(**6**) = **27** and **divide** it by the count 27 / 5 = **5.4** and format it to the **second decimal** point. | |
| **Input Output** | |
| Diamonds|Silver|Shotgun|Gold  Loot Silver Medals Coal  Drop -1  Drop 1  Steal 6  Yohoho! | Coal, Diamonds, Silver, Shotgun, Gold, Medals  Failed treasure hunt. |

## JS Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| (["Gold|Silver|Bronze|Medallion|Cup",  "Loot Wood Gold Coins",  "Loot Silver Pistol",  "Drop 3",  "Steal 3",  "Yohoho!"]) | Medallion, Cup, Gold  Average treasure gain: 5.40 pirate credits. |
| **Comments** | |
| The first command **"Loot Wood Gold Coins"** adds **Wood** and **Coins** to the chest but **omits** Gold since it is already contained. The chest now has the following items:  **Coins Wood Gold Silver Bronze Medallion Cup**  The **second** command adds **only Pistol** to the chest  The **third** command **"Drop 3"** removes the **Gold** from the chest, but immediately adds it at the **end**:  **Pistol Coins Wood Silver Bronze Medallion Cup Gold**  The **fourth** command **"Steal 3"** removes the **last 3** items **Medallion**, **Cup**, **Gold** from the chest and prints them.  In the end calculate the average treasure gain which is the sum of all items length Pistol(**6**) + Coins(**5**) + Wood(**4**) + Silver(**6**) + Bronze(**6**) = **27** and **divide** it by the count 27 / 5 = **5.4** and format it to the **second decimal** point. | |
| **Input Output** | |
| (["Diamonds|Silver|Shotgun|Gold",  "Loot Silver Medals Coal",  "Drop -1",  "Drop 1",  "Steal 6",  "Yohoho!"]) | Coal, Diamonds, Silver, Shotgun, Gold, Medals  Failed treasure hunt. |

# Problem 3 - Man-O-War

Problem for exam preparation for the [Programming Fundamentals Course @SoftUni](https://softuni.bg/courses/programming-fundamentals-csharp-java-js-python).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/Practice/Index/1773#2>.

*The pirates encounter a huge Man-O-War at sea.*

Create a program that **tracks** the **battle** and either chooses a **winner** or prints a **stalemate**. On the **first line,** you will receive the **status** of the **pirate ship**, which is a **string** representing **integer sections** separated by **">"**. On **the second line,** you will receive the **same** type of status, but for the **warship**:

**"{section1}>{section2}>{section3}… {sectionn}"**

On the **third line,** you will receive the **maximum health capacity** a section of the ship can reach.

The following lines represent commands **until** **"Retire"**:

* **"Fire {index} {damage}"** - the pirate ship **attacks** the warship with the **given damage** at that section. Check if the **index is valid** and if not, **skip** the command. If the section **breaks** (health <= 0) the warship **sinks**, print the following and **stop** the program: **"You won! The enemy ship has sunken."**
* **"Defend {startIndex} {endIndex} {damage}"** - the warship **attacks** the pirate ship with the **given damage** at that **range** (**indexes are inclusive)**. Check if both **indexes are valid** and if not, **skip** the command. If the section **breaks** (health <= 0) the pirate ship **sinks**, print the following and **stop** the program:

**"You lost! The pirate ship has sunken."**

* **"Repair {index} {health}"** - the crew **repairs** a section of the **pirate ship** with the **given health**. Check if the **index is valid** and if not, **skip** the command. The health of the section **cannot** exceed the **maximum health capacity**.
* **"Status"** - prints the **count** of all sections of the **pirate ship** that need repair soon, which are all sections that are **lower than 20%** of the **maximum** **health capacity**. Print the following:

**"{count} sections need repair."**

In the end, if a **stalemate** occurs, print the **status** of **both** ships, which is the **sum** of their individual sections, in the following format:

**"Pirate ship status: {pirateShipSum}**

**Warship status: {warshipSum}"**

## Input

* On the **1st line,** you are going to receive the **status** of the **pirate ship** (**integers** separated by **'>'**)
* On the **2nd line,** you are going to receive the **status** of the **warship**
* On the **3rd line,** you will receive the **maximum health** a section of a ship can reach.
* On the following **lines**, until **"Retire"**, you will be receiving commands.

## Output

* Print the output in the **format** **described** **above**.

## Constraints

* The **section numbers** will be integers in the range [**1**….**1000**]
* The **indexes** will be integers [**-200**….**200**]
* The **damage** will be an integer in the range [**1**….**1000**]
* The **health** will be an integer in the range [**1**….**1000**]

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| 12>13>11>20>66  12>22>33>44>55>32>18  70  Fire 2 11  Fire 8 100  Defend 3 6 11  Defend 0 3 5  Repair 1 33  Status  Retire | 2 sections need repair.  Pirate ship status: 135  Warship status: 205 | |
| **Comments** | | |
| First, we receive the command "**Fire 2 11**", and damage the warship at section index 2, which is currently 33, and after reduction, the status of the warship is the following:  **12 22 22 44 55 32 18**  The **second** and **third** commands have **invalid indexes**, so we skip them.  The **fourth** command, **"Defend 0 3 5"** damages **4 sections** of the pirate ship with **5,** which results in the following states:  **7 8 6 15 66**  The **fifth** command, **"Repair 1 33"** repairs the pirate ship section and adds **33 health** to the current **8,** which results in **41**  Only **2 sections** of the pirate ship (**7** and **6**) need repair soon.  In the end, there is a **stalemate,** so we print both ship statuses (**sum** of all sections). | | |
| **Input** | | **Output** |
| 2>3>4>5>2  6>7>8>9>10>11  20  Status  Fire 2 3  Defend 0 4 11  Repair 3 18  Retire | 3 sections need repair.  You lost! The pirate ship has sunken. | |

## JS Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| (["12>13>11>20>66",  "12>22>33>44>55>32>18",  "70",  "Fire 2 11",  "Fire 8 100",  "Defend 3 6 11",  "Defend 0 3 5",  "Repair 1 33",  "Status",  "Retire"]) | 2 sections need repair.  Pirate ship status: 135  Warship status: 205 |
| **Comments** | |
| First, we receive the command "**Fire 2 11**", and damage the warship at section index 2, which is currently 33, and after reduction, the status of the warship is the following:  **12 22 22 44 55 32 18**  The **second** and **third** commands have **invalid indexes**, so we skip them.  The **fourth** command, **"Defend 0 3 5"** damages **4 sections** of the pirate ship with **5,** which results in the following states:  **7 8 6 15 66**  The **fifth** command, **"Repair 1 33"** repairs the pirate ship section and adds **33 health** to the current **8,** which results in **41**  Only **2 sections** of the pirate ship (**7** and **6**) need repair soon.  In the end, there is a **stalemate,** so we print both ship statuses (**sum** of all sections). | |
| **Input** | **Output** |
| (["2>3>4>5>2",  "6>7>8>9>10>11",  "20",  "Status",  "Fire 2 3",  "Defend 0 4 11",  "Repair 3 18",  "Retire"]) | 3 sections need repair.  You lost! The pirate ship has sunken. |