# Exam Preparation – 14 June 2024

## **Worms&Holes**

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/4438#0**](https://judge.softuni.org/Contests/Practice/Index/4438#0)

The first line will give you **a sequence of integers representing worms**. Afterwards, you will be given another **sequence of integers representing holes**.

You have to start with the **last worm** and try to match it with the **first hole**.

* If their **values** are **equal**, the worm fits the hole and can get into it. After that, you should **remove both** of them from their sequences. Otherwise, you should **remove** the **hole** and **decrease** the **value** of the **worm** by **3**.
* If the **worm** value becomes **equal to or below** **0**, **remove it** from the sequence **before** trying to **match** it with the hole.

You need to **stop** **matching** when you have **no more worms or holes**.

### Input / Constraints

* On the **first line**, you will receive the integers, representing the **worm size**, **separated** by a **single space**.
* On the **second line,** you will receive the integers, representing the **hole size**, **separated** by a **single space**.
* All given numbers will be valid integers in the range **[1, 50]**.

### Output

* On the **first** line:
* If there are matches print the following:
  + "**Matches: {matchesCount}"**
* If there are no matches print the following:
* **"There are no matches."**
* On the **second** line print:
* If there are no worms left and **all of them fit a hole**:
  + "**Every worm found a suitable hole!**"
* If there are no worms left but only some of them fit a hole:
  + "**Worms left: none**"
* If there are worms left:
  + "**Worms left: {worm1}, {worm2}, (…),{wormn}**"
* On the **third** line print:
* If there are no holes left:
  + "**Holes left: none**"
* If there are holes left:
  + "**Holes left: {hole1}, {hole2}, (…),{holen}**"

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| **9 5 8 13**  **13 8 5 6** | **Matches: 3**  **Worms left: 6**  **Holes left: none** | The first pair is the **first hole** with a value of 13 and the **last worm** with a value of 13, their **values are equal**, so the worm gets into the hole **and we remove values** from the **sequences**. Next, there are **two more matches** (8 = 8) and (5 = 5) you should **remove both of them**, too. But the value of the **next worm is 9** and the value of the **next hole is 6**, (9 > 6) so we **reduce** the **worm’s value** by 3 and **remove** the **hole**. |
| **17 20 25 25 30**  **9 8 7 21 5 4 3 2 1** | **Matches: 1**  **Worms left: 17, 20, 25, 10**  **Holes left: none** |  |
| **9 8 7 6**  **6 7 8 9** | **Matches: 4**  **Every worm found a suitable hole!**  **Holes left: none** |  |
| **10 10 10 10**  **5** | **There are no matches.**  **Worms left: 10, 10, 10, 7**  **Holes left: none** |  |

# 02. Fishing Competition

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/4355#1**](https://judge.softuni.org/Contests/Practice/Index/4355#1)

*You are a longtime captain of an old fishing vessel. The new fishing season begins and you prepare your ship to set sail in search of the big catch…*

You will be given an integer **n** for the **size** of the **fishing area** with a **square** shape. On the next **n** lines, you will receive the **rows** of the **fishing area**. You will be placed in a **random position**, marked with the letter '**S**'. There will be fishing passages on **random positions**, marked with a **single digit**. There will be whirlpools **marked** with '**W**'. All of the empty **positions** will be marked with **'-'**.

Each turn until the "**collect the nets**" command is received you will be given **commands** for **your movement**. Move commands will be: "**up**", "**down**", "**left**", and "**right**".

* If you **move** to a **fish passage**, you **collect** **the amount** **equal** to the **digit** **there**, the **passage disappears** and should be replaced by **'-'**.
* If you fall into a whirlpool – the ship sinks and loses its catch, the program ends.
* If you leave the fishing area (go out of the boundaries of the matrix) depending on the move command you will be moved to the opposite side of the one you were on.

/**Example:** In a 3x3 matrix you are at position **[1,2]** and receive the command "**right**" you will be moved to position **[1,0]**./

You need **at least** **20 tons** of fish to be considered a successful season. Keep in mind that even if the **quota is reached** the **ship continues to move**.

### Input

* On the first line, you are given the integer **n** – the size of the **square** matrix.
* The **next n lines** hold the values for every **row**.
* On each of the next lines, you will get a move command.

### Output

* On the first line:
* If the ship **falls into a whirlpool**, print only this message and stop the program:
* **"You fell into a whirlpool! The ship sank and you lost the fish you caught. Last coordinates of the ship: [n,n]"**
* If the **ship reaches** the quota:
* **"Success! You managed to reach the quota!"**
* If the **ship did not reach** the quota:
* **"You didn't catch enough fish and didn't reach the quota!**

**You need {lack of fish} tons of fish more."**

* On the **next** lines.
* If the catch quantity is bigger than zero, print:
* **"Amount of fish caught: {quantity} tons."**

else: **do not print** anything.

* If you **didn't get into a whirlpool**, print the **matrix**.

### Constraints

* The size of the **square** matrix will be between **[2…10].**
* Only the letters '**S**' and '**W**' will be present in the matrix.
* The **fish passages** are represented by **single positive digits** /tons/ between **[1…9]**.
* It is expected that there will only be either **zero** or **one** **whirpool** present, marked with the **letter** - '**W**'.
* Your position will be marked with '**S**'.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  ---S  ----  9-5-  34--  down  down  right  down  collect the nets | You didn't catch enough fish and didn't reach the quota! You need 8 tons of fish more.  Amount of fish caught: 12 tons.  ----  ----  --5-  S4-- |
| **Comment** | |
| The first command is **"down"**. The ship moves to position **[1,3]** followed by the command **"down" [2,3]** andthen the command **"right".** The ship leaves the matrix's boundaries and transfers to the opposite side at position **[2,0]**. The ship comes across a **fish passage** with a quantity of 9 tons and gets it. After executing the third command, the **fishing area** will appear as follows:  ----  ----  S-5-  34--  Then you receive the command **"down"** again.You move to the passage of **3** tons and add them to the others **9**. Your catch is **9 + 3 = 12** tons. In the end, you get the command **"collect the nets"** and the program ends. | |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  S---9  777-1  W333-  11111  -----  down  down  right  down  collect the nets | You fell into a whirlpool! The ship sank and you lost the fish you caught. Last coordinates of the ship: [2,0] |
| **Comment** | |
| The first command is **"down"**. The ship moves to position **[1,0]** and gets **7** tons of fish. Follow the command **"down"** -> **[2,0]** The ship **falls into a whirlpool and sinks.** You lose the entire catch and the program ends. | |
| **Input** | **Output** |
| 5  S---9  777-1  --5--  11W11  988--  down  down  down  down  down  down  right  right  right  collect the nets | Success! You managed to reach the quota!  Amount of fish caught: 31 tons.  ----9  ---S1  --5--  -1W11  -88-- |
| **Comment** | |
| Result is: **7 + 1 + 9 +7 + 7 = 31**. You succeeded! | |

# 03. Spacecraft

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/4547#2**](https://judge.softuni.org/Contests/Practice/Index/4547#2)

*Spacecraft are vehicles designed to travel and operate in outer space. They serve various purposes, including scientific exploration, communication, navigation, weather monitoring, and satellite deployment. These incredible machines have enabled humanity to reach beyond the confines of Earth and explore the vast expanse of the cosmos.*

### **Preparation**

Download the skeleton provided in Judge. **Do not** change the **packages**!

**Pay attention to name the package spaceCrafts, all the classes, their fields, and methods the same way they are presented in the following document. It is also important to keep the project structure as described.**

### **Problem description**

Your task is to create a repository that stores spacecraft by creating the classes described below.

### **Spacecraft**

First, write a class **Spacecraft** with the following properties:

* **name: String**
* **missionType: String**
* **target: String**
* **objective: String**
* **weight: int**

The class **constructor** should receive **name, missionType, target, objective** and **weight**. You need to create the appropriate **getters and setters**. All spacecraft names will be **unique.** It is guaranteed that there **will be no duplicates** of names.

Override the **toString()** method in the following format:  
**The mission of {name} was to reach {target} and to {objective}**

### **LaunchPad**

**Next**, write a class **LaunchPad**. The **LaunchPad** class should have those **properties**:

* **name: String**
* **capacity: int**
* **spacecrafts: List<Spacecraft>**

The class **constructor** should receive **name** and **capacity.** Also, it should initialise the **spacecrafts** with a new **collection** instance.Implement the following features:

* **Method addSpacecraft(Spacecraft spacecraft)** – **adds** an **entity** to the data **if** a spacefor it, otherwise print: **"This launchpad is at full capacity!"**
* **Method removeSpacecraft(String name)** – removes a **Spacecraft** by **given name,** if such **exists**, and **returns boolean** (true if it is removed, otherwise – false)
* **Method getHeaviestSpacecraft()**– **returns String** the **heaviest spacecraft** by **weight** in the given launchpad in the following format:
* **"{spaceCraft name} - {spaceCraft weight}kg."**
* **Method getSpacecraft(String name)** – **returns** the **spacecraft** with the **given name,** otherwise – returns **null**
* **Method getCount()** – **returns** the **count** of **spacecrafts** in the given launchpad
* **Method getSpacecraftsByMissionType(String missionType)** **–** **returns** **List** **–** a **collection** of **Spacecraft** whichholds the spacecrafts having the **same** **missionType** in the given launchpad.

In case there are **no** spacecrafts to **respond** to this **condition** (missionType) print:

* **"There are no spacecrafts to respond this criteria."**
* **Method getStatistics()** – **returns** a **String** in the following **format** (print the spacecrafts in **order of addition**):
  + **"Spacecrafts launched from {launchpad name}:  
    1. {spacecraft name}  
    2. {spacecraft name}  
     (…)**

1. **{spacecraft name}"**

If there are no spacecrafts in some launchpad print:

* **"Spacecrafts launched from {launchpad name}:**

**none"**

### **Constraints**

* The **name** and **weight** of the spacecraft will always be **unique**.
* You will always have a spacecraft added before receiving methods manipulating the Launchpad's spacecraft.

### **Examples**

This is an example of how the **Launchpad** class is **intended to be used**.

|  |
| --- |
| **Sample code usage** |
| //Initialize the repositories (LaunchPad)  LaunchPad kennedySC = **new** LaunchPad("Kennedy Space Center", 3);  LaunchPad capeCanaveralAFS = **new** LaunchPad("Cape Canaveral Air Force Station", 5);  //Initialize entities (Spacecraft)  Spacecraft galileo = **new** Spacecraft("Galileo", "Orbiter", "Europa", "study Jupiter and its mysterious moons", 223);  Spacecraft cassini = **new** Spacecraft("Cassini-Huygens", "Orbiter", "Saturn", "analyze the composition and atmosphere of Saturn", 2523);  Spacecraft magellan = **new** Spacecraft("Magellan", "Orbiter", "Venus", "image the entire surface of Venus", 3445);  Spacecraft huygens = **new** Spacecraft("Huygens", "Atmospheric Probe/Lander", "Titan", "provide a detailed study of Titan's atmosphere", 318);  Spacecraft voyager1 = **new** Spacecraft("Voyager-1", "Multiple Flybys", "Outer Solar System", "chart the edge of interstellar space", 733);  //Add Spacecraft  kennedySC.addSpacecraft(galileo);  kennedySC.addSpacecraft(cassini);  kennedySC.addSpacecraft(magellan);  kennedySC.addSpacecraft(huygens); //This launchpad is at full capacity!  capeCanaveralAFS.addSpacecraft(voyager1);  //Get Spacecraft By Mission Type  kennedySC.getSpacecraftsByMissionType("Orbiter").forEach(spacecraft -> System.***out***.println(spacecraft.getName()));  //Galileo  //Cassini-Huygens  //Magellan  capeCanaveralAFS.getSpacecraftsByMissionType("Multiple Flybys").forEach(spacecraft -> System.***out***.println(spacecraft.getName()));  //Voyager-1  //Remove Spacecraft  System.***out***.println(capeCanaveralAFS.removeSpacecraft("Voyager-1")); //true  //Launchpad Get Count Of Spacecrafts  System.***out***.println(capeCanaveralAFS.getCount()); //0  //Spacecraft toString()  System.***out***.println(galileo);  //The mission of Galileo was to reach Europa and to study Jupiter and its mysterious moons  //Get Heaviest Spacecraft  System.***out***.println(kennedySC.getHeaviestSpacecraft()); //Magellan - 3445kg.  //LaunchPad Get Statistics  System.***out***.println(kennedySC.getStatistics());  //Spacecrafts launched from Kennedy Space Center:  //1. Galileo  //2. Cassini-Huygens  //3. Magellan  System.***out***.println(capeCanaveralAFS.getStatistics());  //Spacecrafts launched from Cape Canaveral Air Force Station:  //none |

### **Submission**

Submit **single .zip file**, containing **spaceCrafts** package, **with the classes inside** (**Launchpad**, **Spacecraft** and the **Main** **class)**, there is no specific content required inside the **Main** class e.g. you can do any kind of local testing of your program there. However, there should be **main(String[] args)** method inside.