# Exam Preparation – 20 June 2022

## Flower Wreaths

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2531#0**](https://judge.softuni.org/Contests/Practice/Index/2531#0)

*You want to go on a flowers wreath competition but to participate you have to make at least 5 flower wreaths.*

You will be given **two sequences of integers, representing roses and lilies**. You need to start making wreaths **knowing that one wreath needs 15 flowers**. Your goal is to make **at least 5 flower wreaths**.

You will start crafting from the **last lilies** and the **first roses**. If the **sum** of their values is **equal** to **15** – **create one wreath** and **remove** them. If the sum is **bigger than 15**, just **decrease** the value of the **lilies** **by 2.** If the sum is **less than 15 you have to store them for later and remove them**. You need to **stop** combining when you have **no more roses or lilies**. In the end, if you have **any stored flowers you should make as many wreaths as you can with them**.

### Input

* On the **first line**, you will receive the integers representing the **lilies**, **separated** by **", "**.
* On the **second line**, you will receive the integers representing the **roses**, **separated** by "**,** ".

### Output

* Print whether you have succeeded making **at least 5 wreaths**:
  + **"You made it, you are going to the competition with {count of wreaths} wreaths!"**
  + **"You didn't make it, you need {wreaths needed} wreaths more!"**

### Constraints

* All of the given numbers will be valid integers in the range **[0, 120]**.
* Don't have situation with negative number.

### Examples

|  |  |  |
| --- | --- | --- |
| ****Input**** | ****Output**** | |
| **10, 15, 2, 7, 9, 13**  **2, 10, 8, 12, 0, 5** | **You made it, you are going to the competition with 5 wreaths!** | |
| ****Comment**** | | |
| We start with the last lilies (13) and the first roses (2) -> 13 + 2 = 15 -> 15 = 15 So we create one wreath and remove them bouth.  Next we have 9 + 10 = 19 -> 19 > 15 so we decrease the lilies by 2 -> 7 + 10 = 17 and we decrease the liles by 2 -> 5 + 10 = 15 and we create one more wreath and remove them.  Next, we have 7 + 8 = 15. We create one more wreaht and remove them.  Next, we have 2 + 12 = 14 -> 14 < 15 so we have to store theire sum for later and remove theme.  Next, we have 15 + 0 = 15 so we create one more wreath.  And last we have 10 + 5 = 15, we create one more wreath and stop mixing because we don’t have any flowers left.  Now we have a tottal of 5 wreaths and we also have 14 flowers left but we cant create wreath because 14 < 15. | | |
| ****Input**** | | ****Output**** |
| **10, 5, 3, 7, 8**  **5, 10, 8, 7, 6** | | **You didn't make, you need 1 wreaths more!** |
| Comment | | |
| We start with 8 + 5 = 13 -> 13 < 15 -> we have to store their sum for later and remove them.  Next, we have 7 + 10 = 17 -> we decrease the lilies by 2 -> 5 + 10 = 15 -> 15 = 15 and we create one wreath.  Next, we have 3 + 8 = 11 -> 11 < 15 -> we store their sum for later and remove them.  Next, we have 5 +7 = 12 -> we store their sum for later and remove them.  Next, we have 10 + 6 = 16 -> 16 > 15 we decrease the lilies by 2 -> 8 + 6 = 14 and we store their sum for later and remove them.  We stop crafting because we don’t have any flowers left and we have 1 wreath and 50 stored flowers. We create 3 more wreaths because 3 \* 15 = 45 -> 50 – 45 = 5 -> 5 < 15. | | |

## Bomb

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2805#1**](https://judge.softuni.org/Contests/Practice/Index/2805#1)

One of the mission that new agents have to complete is called BombField. Your task is to implement the mission into a simple program.

We get as input **the size** of the **field** in which our sapper moves. The field is **always a square**. After that we receive the commands wich represent the directions in which the sapper should move. The sapper **starts** from **s**-position The commands will be: **left/right/up/down.** If the sapper reaches the side edge of the field (left, right,up or down), it **remains on his current possition**. The possible characters that may appear on the screeen are:

* **+** – regular position on the field.
* **e** – end of the route.
* **B** – bomb
* **s** – the place where the **sapper starts**

Each time when sapper finds a bomb, he deactivates it, and **replace "B" with "+"**. Keep track of the **count of the bombs**. Each time you find a bomb, you have to print the following message: **"You found a bomb!".** If sapper **steps at the end of the route game is over (the program stops)** and you have to print the output as shown in the output section. After executing all of the commands there are only 2 possible outcomes (there are not going to be more cases):

* if you found all bombs – you win and the game ends
* if you reach the end point ("e"), you have to stop

Print the corresponding output depending on the case.

## Input

* **Field size** – an integer number.
* **Commands to move** the sapper – an array of strings separated by **","**.
* **The field: some of the following characters (+, e, B, s),** separated by whitespace (" ");

## Output

* There are three types of output:
  + If all of the bombs have cleared print the following output: "**Congratulations! You found all bombs!**"
  + If you reached the end, you have to stop moving and print the following line: **"END! {bombs left} bombs left on the field"**
  + If there are no more commands and none of the above cases happens, you have to print the following message: **"{bombs left} bombs left on the field. Sapper position: ({row},{col})"**

## Constraints

* The **input numbers** will be a 32-bit integer in the range [0 … 2 147 483 647].
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5  up,right,right,up,right  + + + B +  + + + e +  + + B + +  s + + B +  + + B + + | You found a bomb!  END! 3 bombs left on the field | After executing all of the commands,sapper move to the end point.But there are some boms left, so we print the information. |
| 4  up,right,right,right,down  + + + e  + + B +  + s + B  + + + + | You found a bomb!  You found a bomb!  Congratulations! You found all bombs! | The sapper reached the end of the field, so he remains hes position and we print the message. After all the bombs are found, we should stop the program and print the appropriate message. |

## Hotel

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/3203#2**](https://judge.softuni.org/Contests/Practice/Index/3203#2)

## Preparation

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

**Pay attention to name the package hotel, 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 people by creating the classes described below.

**Person**

First, write a Java class **Person** with the following fields:

* **name: String**
* **id: int**
* **age: int**
* **hometown: String – "n/a" by default**

The class **constructor** should receive **name, id, age, hometown**. You need to create the appropriate **getters and setters**. Override the **toString()** method in the following format:

**"Person {name}: {id}, Age: {age}, Hometown: {hometown}"**

**Hotel**

**Next**, write a Java class **Hotel** that has **a roster** (a collection that stores **Person** entities). All entities inside the repository have the **same fields**. Also, the **Hotel** class should have those **fields**:

* **name: String**
* **capacity: int**

The class **constructor** should receive **name** and **capacity**, also it should initialize the **roster** with a new instance of the collection.Implement the following features:

* Method add(Person person) - **adds** an **entity** to the roster **if** **there** **is** **room** for it
* Method remove(String name) - removes a person by **given name,** if such **exists**, and **returns boolean**
* Method **getPerson(String name, String hometown)** – returns the people with the **given name** and **hometown** or **null** if there is no such person.
* Getter getCount() – **returns** the **number** of people.
* **getStatistics()** – **returns** a **String** in the following **format**:
  + **"The people in the hotel {name} are:  
    {Person1}  
    {Person2}  
    (…)**"

## Constraints

* The **names** of the people will be **always unique**.
* You will always have a person added before receiving methods manipulating the Hotel's people.

## Examples

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

|  |
| --- |
| Sample code usage |
| package hotel;  public class Main {  public static void main(String[] args) {  //Initialize the repository (hotel)  Hotel hotel = new Hotel("Hilton", 30);  //Initialize entity  Person person = new Person("Mark", 11111, 41, "Sofia");  //Print person  System.*out*.println(person);  //Person Mark: 11111, Age: 41, Hometown: Sofia   //Add person  hotel.add(person);  System.*out*.println(hotel.getCount()); //1  System.*out*.println(hotel.remove("Anna")); //false   Person firstPerson = new Person("Alice", 22121, 18, "London");  Person secondPerson= new Person("Lizzy", 31311, 24, "Varna");  Person thirdPerson = new Person("Lucy", 54122, 31, "Birmingham");  Person fourthPerson = new Person("Maria", 66611, 41, "Sofia");    //Add people  hotel.add(firstPerson);  hotel.add(secondPerson);  hotel.add(thirdPerson);  hotel.add(fourthPerson);  // Get person  Person personForGet = hotel.getPerson("Lucy", "Birmingham");  Person personForGet1 = hotel.getPerson("Anna", "Burgas");  System.out.println(personForGet); // Person Lucy: 54122, Age: 31, Hometown: Birmingham  System.out.println(personForGet1); // null  //Count  System.out.println(hotel.getCount()); // 5  //Remove Person  System.*out*.println(hotel.remove("Alice")); //true  // Get Statistics  System.out.println(hotel.getStatistics());  // The people in the hotel Hilton are:  // Person Mark: 11111, Age: 41, Hometown: Sofia  // Person Lizzy: 31311, Age: 24, Hometown: Varna  // Person Lucy: 54122, Age: 31, Hometown: Birmingham  // Person Maria: 66611, Age: 41, Hometown: Sofia  } } |