# Santa's Present Factory



*Christmas is just around the corner and this year Santa has decided to share his secret with you. Get ready to learn how his dwarf's craft all the presents.*

First, you will receive a sequence of **integers**, representing the number of materials for crafting toys in one box. After that, you will be given another sequence of **integers** – their magic level.

|  |  |
| --- | --- |
| **Present** | **Magic needed** |
| Doll | 150 |
| Wooden train | 250 |
| Teddy bear | 300 |
| Bicycle | 400 |

Your task is to **mix** materials with magic so you can craft presents, listed in the table below with the **exact** magic level.

To craft a toy, you have to take the **last box with materials** and the **first magic level value**. The total magic level is calculated by their multiplication. If the result **equals** one of the levels described in the table above, you craft the present and **remove both** materials and magic value. **Otherwise**:

* If the product of the operation is a **negative** **number**, then you have to sum the values together, remove them both from their positions and the result should be added to the materials.
* If the product **doesn’t equal** one of the magic levels in the table and is a **positive** number, remove only the magic value and **increase** the material value by **15**.
* If the magic or material (or both) **equals** **0**, remove it (or both) and continue crafting the presents.

Stop crafting presents when you **run out** of boxes of materials **or** magic level values.

Your task is considered done if you manage to craft either one of the pairs - **a doll and a train** **or a teddy bear and a bicycle**.

## Input

* The first line of input will represent the values of boxes with materials - **integers**, separated by a **single space.**
* On the second line, you will be given the magic values - **integers** again, separated by a **single space.**

## Output

* On the first line - print whether you've succeeded in crafting the presents
* **"The presents are crafted! Merry Christmas!"**
* **"No presents this Christmas!"**
* On the next two lines print the **materials** and **magic** that are **left**, **if there are any**, **otherwise skip the line**
  + "**Materials left: {material1}, {material2}, …**"
  + "**Magic left: {magicValue1}, {magicValue2}, …**
* On the next lines print the presents you **have crafted at least once,** ordered **alphabetically** in the format:

**"{toy name}: {amount}"**

**…**

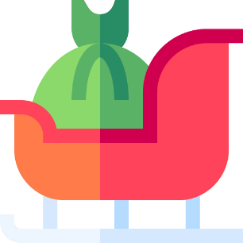
## Constraints

* All of the materials' values will be **integers** in the range **[1, 100].**
* Magic level values will be **integers** in the range **[-10, 100].**
* In all cases, at least one present will be crafted.

## Examples

|  |  |  |
| --- | --- | --- |
| ****Input**** | ****Output**** | ****Comment**** |
| **10 -5 20 15 -30 10**  **40 60 10 4 10 0** | **The presents are crafted! Merry Christmas!**  **Materials left: 20, -5, 10**  **Bicycle: 1**  **Teddy bear: 2** | First, we have 40\*10=400 which is the needed magic for a bicycle. Remove both. 60\*(-30) = -1800 (negative). 60+(-30)=30. Remove 60 and -30. Add 30 to materials. 30\*10=300 (bear). Remove both. 4\*15=60, so remove 4 and the material is increased by 15 (15+15=30). 10\*30=300 (bear). Print desired text. |
| **30 5 15 60 0 30**  **-15 10 5 -15 25** | **No presents this Christmas!**  **Materials left: 20, 30**  **Doll: 1**  **Teddy bear: 1** |  |

# Present Delivery



*Now that the presents are ready, Santa has to deliver them to the kids.*

You will receive an integer **m** for the **count** of **presents** Santa has and an integer **n** for the **size** of the **neighborhood** with a **square** shape. On the next lines, you will receive the **matrix**, which represents the neighborhood.

Santa will be in a **random** **cell**, marked with the letter '**S**'. Each cell stands for a house where children may live. If the cell has '**X**' on it, that means there lives a **naughty** kid. Otherwise, if a **nice** kid lives there, the cell is marked by '**V**'. There can also be cells marked with '**C**'for cookies. **All of the empty positions** will be marked with **'-'**.

Santa can move "**up**", "**down**", "**left**", "**right**". These will be the **commands** that you receive. If he moves to a house with a **nice** kid, the kid **receives a present**, but if Santa reaches a house with a **naughty** kid, he **doesn’t** drop a present. If the command sends Santa to a cell marked with '**C**', Santa eats cookies and becomes happy and extra generous so **all the kids around him**\* receive presents (doesn’t matter if naughty or nice). If Santa has been to a house and the kid there has received a present, the cell becomes **'-'**.

**Note**: \*around him means on his left, right, upwards, and downwards by one cell. In this case, **Santa** doesn't move to these cells or if he does, he **returns** to the **cell** where the **cookie** was.

If Santa runs out of presents or you receive the command "**Christmas morning**", then you have to end the program.

Keep in mind that you have to check whether all of the nice kids received presents.

### Input

* On the first line, you are given the integer **m** – the count of presents.
* On the second – integer **n** – the size of the neighborhood.
* The **next n lines** hold the values for every **row.**
* On each of the next lines, you will get a command.

### Output

* On the first line:
  + If Santa goes out of the neighborhood, print: "**Santa ran out of presents.**"
* Next print the matrix.
* In the end print one of these messages:
  + If he manages to give **all** the nice kids presents, print:  
    "**Good job, Santa! {count nice kids} happy nice kid/s.**"
  + Otherwise print:   
    **"No presents for {count nice kids} nice kid/s."**

### Constraints

* The size of the **square** matrix will be between **[2…10].**
* Santa’s position will be marked with '**S**'.
* There will **always** be **at** **least** **1** **nice** kid.
* There **won't be a case** where the cookie is on the border of the matrix.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5  4  - X V -  - S - V  - - - -  X - - -  up  right  down  right  Christmas morning | - - - -  - - - S  - - - -  X - - -  Good job, Santa! 2 happy nice kid/s. | Santa has 5 presents. The size of the matrix is 4. After we receive the matrix, we start reading commands. The first one is "up". The "X" means there is a naughty kid, so Santa moves on without dropping any presents. Next, he reaches a nice kid and drops a present. The "down" command moves Santa to an empty cell. The last command before the "Christmas morning" message is "right". Again we have a nice kid. The count of nice kids reached 2 and we don't have any nice kids without presents left. So we print the appropriate message. |
| 1  3  - X V  V S -  V - -  left | Santa ran out of presents!  - X V  S - -  V - -  No presents for 2 nice kid/s. |  |

# Christmas

A picture containing LEGO, vector graphics

Description automatically generated

*Can you imagine Santa without his special bag that carries presents for so many children? Neither can we. So let's have a quick sneak peek inside it and help Santa rearrange it.*

## Preparation

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

**Pay attention to the name of the package (christmas), 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 above.**

## Problem description

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

### Present

First, write a Java class Present with the following fields:

* **name: String**
* **weight: double**
* **gender: String**

The class **constructor** should receive (**name, weight, and gender**).

The class also should have the methods:

* getName()
* getWeight()
* getGender()
* Override the **toString()** method in the following format:

**"Present {name} ({weight}) for a {gender}"**

**Note: Format the weight to the second digit after the decimal point!**

### Bag

**Next**, write a **Java** class Bag that has **data** (a collection that stores the entity **Present**). All entities inside the repository have the **same fields**. Also, the Bag class should have those fields:

* **color:** String
* **capacity:** int
* **data:** List<Present> **-** holds all added presents in the bag

The class **constructor** should receive (**color**, **capacity**), also it should initialize the **data** with a new instance of the collection**.**

Implement the following features:

* getColor()
* getCapacity()
* count() method– **returns** the **number** of presents
* add(Present present) method – **adds** an **entity** to the data **if** **there** **is** **room** for it
* remove(String name) method – removes a present by **given name,** if such **exists**,   
  and **returns boolean**
* heaviestPresent() method – returns the **heaviest** present
* getPresent(String name) method – returns the present with the **given name**
* report() method – **returns** a **string** in the following **format** (print the presents in order of appearance):
  + **"{color of Bag} bag contains:  
    {Present1}  
    {Present2}  
    (…)**"

## Constraints

* The **names** of the presents will be **always unique**.
* The **weights** of the presents will always be with **positive values**.
* You will always have a present added before receiving methods.

## Examples

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

|  |
| --- |
| Sample code usage |
| *// Initialize the repository* Bag bag = **new** Bag(**"black"**, 10);  *// Initialize entity* Present present = **new** Present(**"Doll"**, 0.4, **"girl"**);  *// Print Present* System.***out***.println(present); *// Present Doll (0.40) for a girl*  *// Add Present* bag.add(present);  *// Remove Present* bag.remove(**"Toy"**); *// false*  Present secondPresent = **new** Present(**"Train"**, 2, **"boy"**); *// Add Present* bag.add(secondPresent);  Present heaviestPresent = bag.heaviestPresent();  System.***out***.println(heaviestPresent);  *// Present Train (2.00) for a boy*  Present p = bag.getPresent(**"Doll"**);  System.***out***.println(p);  *// Present Doll (0.40) for a girl*  System.***out***.println(bag.count()); *// 2* System.***out***.println(bag.report());  *// Black bag contains: // Present Doll (0.40) for a girl // Present Train (2.00) for a boy* |

## Submission

Submit a **single .zip file**, containing a **christmas package, with the classes inside (Present, Bag, 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 a **main(String[] args)** method inside.