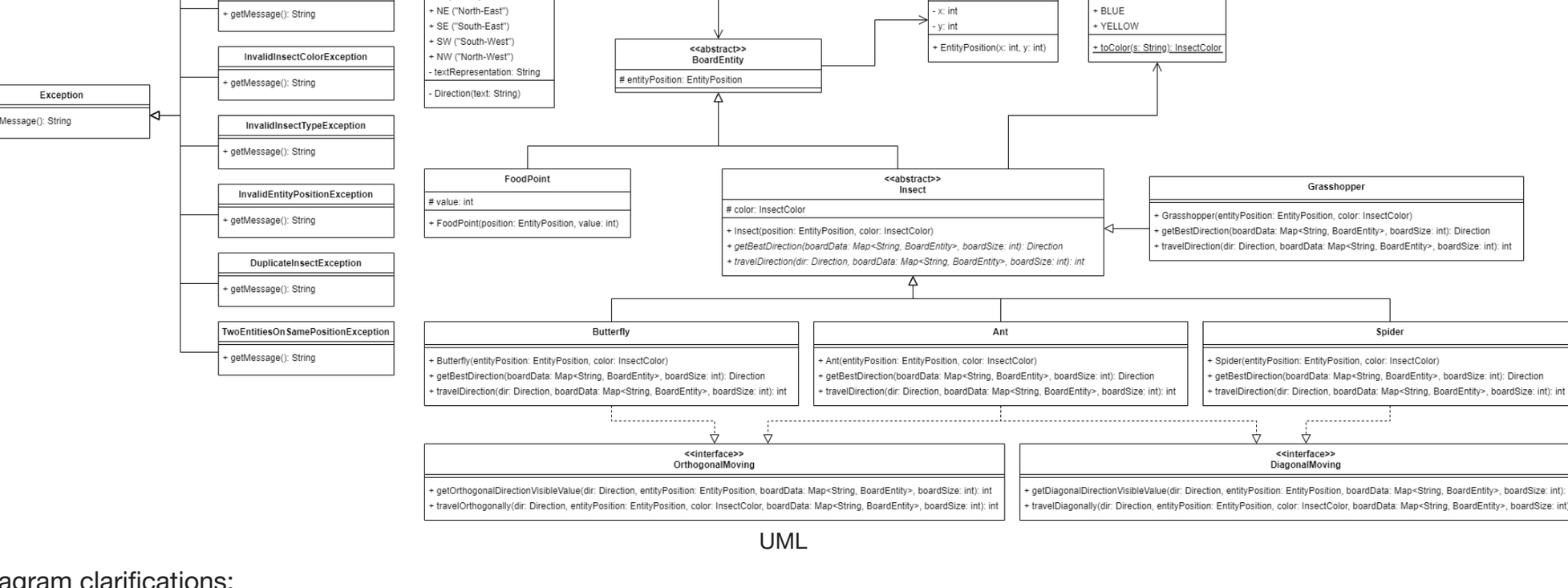
A. Assignment 4

time limit per test: 1 second
memory limit per test: 256 megabytes
input: input.txt
output: output.txt

Your task in this assignment is to implement a simulation of the insects moving game. There are four types of insects that should be considered:

- **Ants** can move vertically, horizontally, and diagonally.
- **Butterflies** can move only vertically and horizontally.
- **Spiders** can move only diagonally.
- **Grasshoppers** can jump only vertically and horizontally but by skipping odd fields.

Your code is required to have at least all elements presented in the given UML Class Diagram, but you are allowed to extend it with additional classes and relations.



UML

Diagram clarifications:

- Method `travelDirection()` is used to simulate insect traveling a specific direction. Therefore, it will return amount of food eaten during the travel, but also, it will update board according to the move of the insect (remove insect and eaten food from the board).
- Method `travelDiagonally()` is used to simulate insect traveling a specific diagonal direction (*North-East, South-East, South-West, North-West*). This method will return amount of eaten food and update the board.
- Method `travelOrthogonally()` is used to simulate insect traveling a specific orthogonal direction (*North, East, South, West*). This method will return amount of eaten food and update the board.

Type of moving for each insect is presented in Figure 1. Each movement of any insect provides visiting all the cells in the chosen direction until the end of the board or until it is killed.

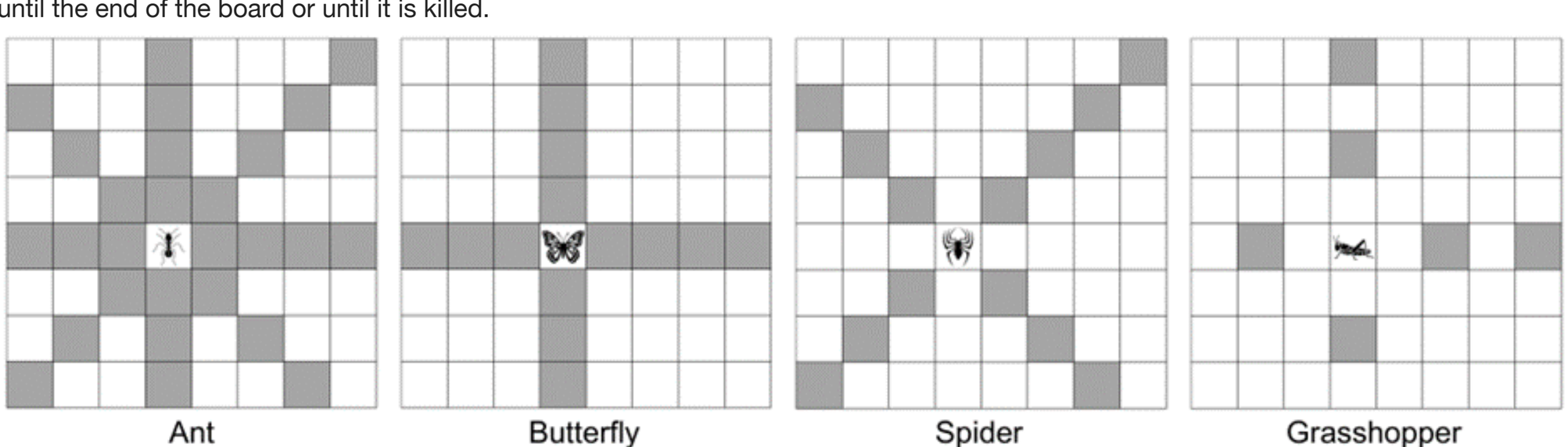


Figure 1. *Insect movements*

In addition to insects, the board can contain the food points with the specific amount of food. For the purpose of this game we will consider that all insects eat the same type of food.
The goal of each insect is by collecting as much food as possible to try to leave the board. However, each insect can choose only one direction and move only in this direction for the rest of time.

- **Ants** can choose *North* (N), *East* (E), *South* (S), *West* (W), *North-East* (NE), *South-East* (SE), *South-West* (SW), and *North-West* (NW) directions.
- **Butterflies** can choose *North* (N), *East* (E), *South* (S), *West* (W) directions.
- **Spiders** can choose *North-East* (NE), *South-East* (SE), *South-West* (SW), and *North-West* (NW) directions (since they can move only diagonally)
- **Grasshoppers** can choose *North* (N), *East* (E), *South* (S), *West* (W) directions.

In addition, each insect is colored in one of the following colors: *Red*, *Green*, *Blue*, *Yellow*.
The example of the starting board for our game is presented in Figure 2. Note that insects are **colored** and that fields with numbers represent the *food points* with the specific amount of food.

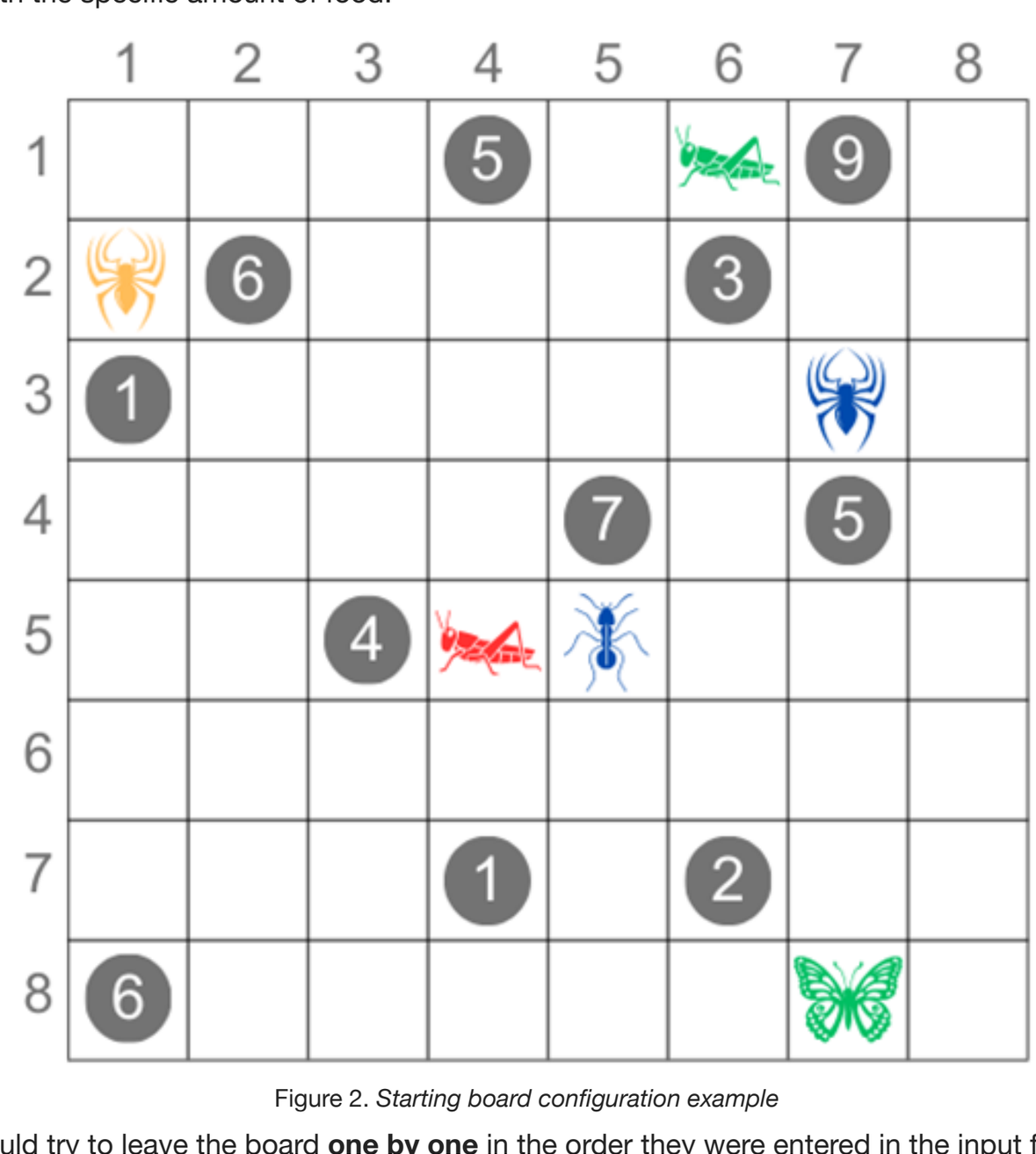


Figure 2. *Starting board configuration example*

In this game the insects should try to leave the board **one by one** in the order they were entered in the input file. For example, if red grasshopper is the first in the input file, it will choose direction and move in this direction, while all other insects will **not move** until red grasshopper leaves the board or dies trying to do it.
The rules of the game are the following:

- Each insect can see all the *food points* on the board, but it cannot see any other insects (even in neighbor cells). Therefore, insects will make decisions based only on positions of the food points, but it will ignore other insects' positions when making decisions. Note that not being able to see other insects can cause **suboptimal decisions** (unexpected deaths).
- Once an insect chooses a direction, it will start moving in this direction (without changing direction) and will eat all food on its path until it is out of the board or it is killed.
- If an insect meets another insect of the same color on its path they will just ignore each other. However, if it visits the cell of differently colored insects, the former will be **killed** by the latter. Note that a **grasshopper will not be killed** if it jumps over the insect of a different color.

The tricky part of the game is that an insect will choose movement direction based only on maximization of eaten food from *food points* positions. However, since insects cannot see other insects, it can happen so that an insect will visit the cell of the different colored insect and will be killed without actually eating all the food on the unvisited cells of the remaining path.
If there are two or more directions with the same amount of food, an insect will prioritize the directions in the following order:

1. *North* (N)
2. *East* (E)
3. *South* (S)
4. *West* (W)
5. *North-East* (NE)
6. *South-East* (SE)
7. *South-West* (SW)
8. *North-West* (NW)

For example, if a red ant sees the same amount of food in the *South-East* (SE) direction and the *North-West* (NW) direction, it will choose the *South-East* (SE) direction because the latter has a **higher priority**.
In addition to the rules, there are some constraints to be considered:

- One cell can contain only one insect, only one *food point* of any amount not less than 1, or nothing.
- There can be only one instance of the specific insect type of the specific color on the board.

Example Game Simulation
Let us say that we have the following initial board configuration.

Insects:

- [5, 4] Red Grasshopper
- [8, 7] Green Butterfly
- [5, 5] Blue Ant
- [3, 7] Blue Spider
- [1, 6] Green Grasshopper
- [2, 1] Yellow Spider

Food points:

- 5 [1, 4]
- 9 [1, 7]
- 6 [2, 2]
- 3 [2, 6]
- 1 [3, 1]
- 7 [4, 5]
- 5 [4, 7]
- 4 [5, 3]
- 1 [7, 4]
- 2 [7, 6]
- 6 [8, 1]

Given board configuration is represented in the following picture.

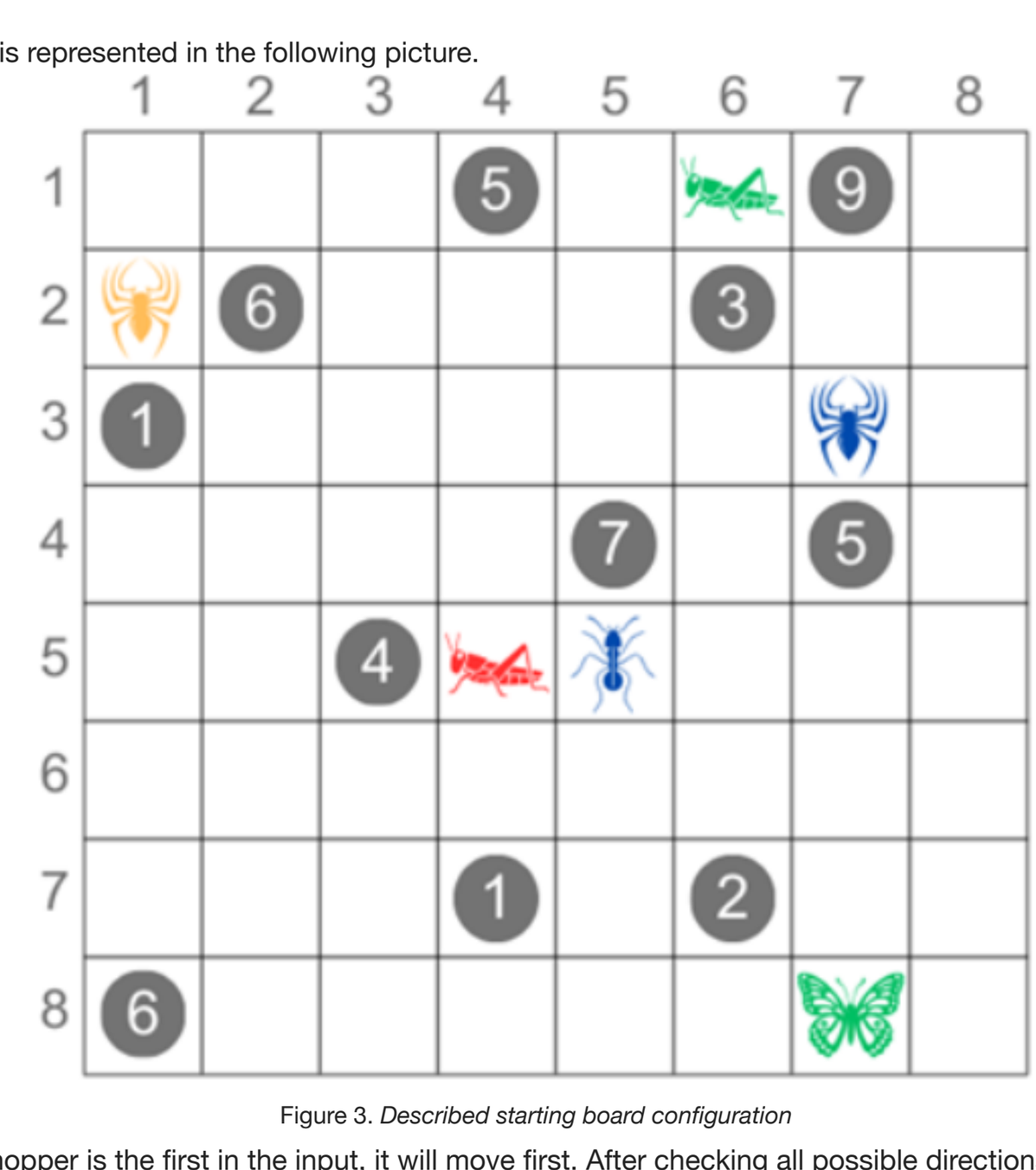
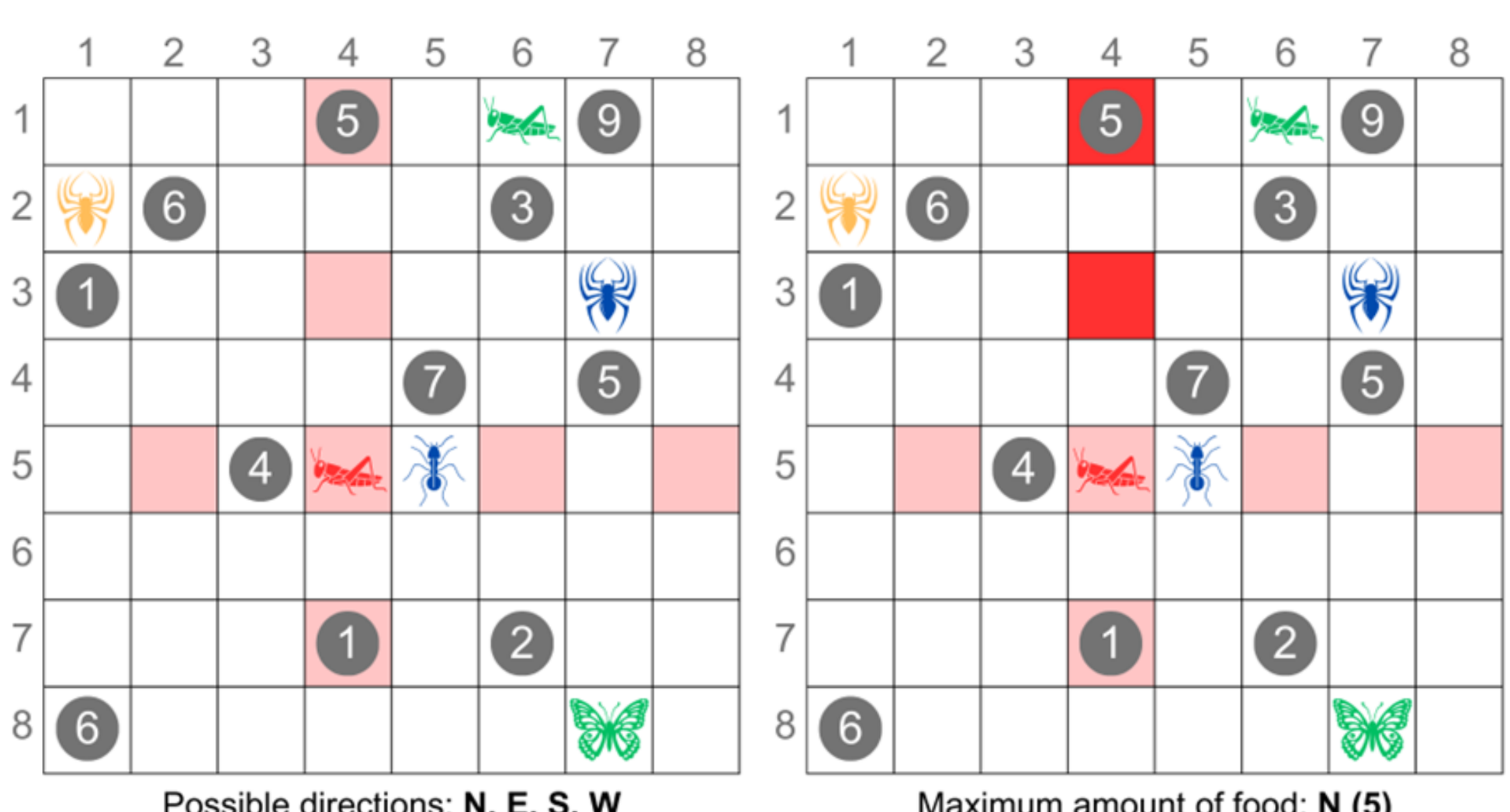
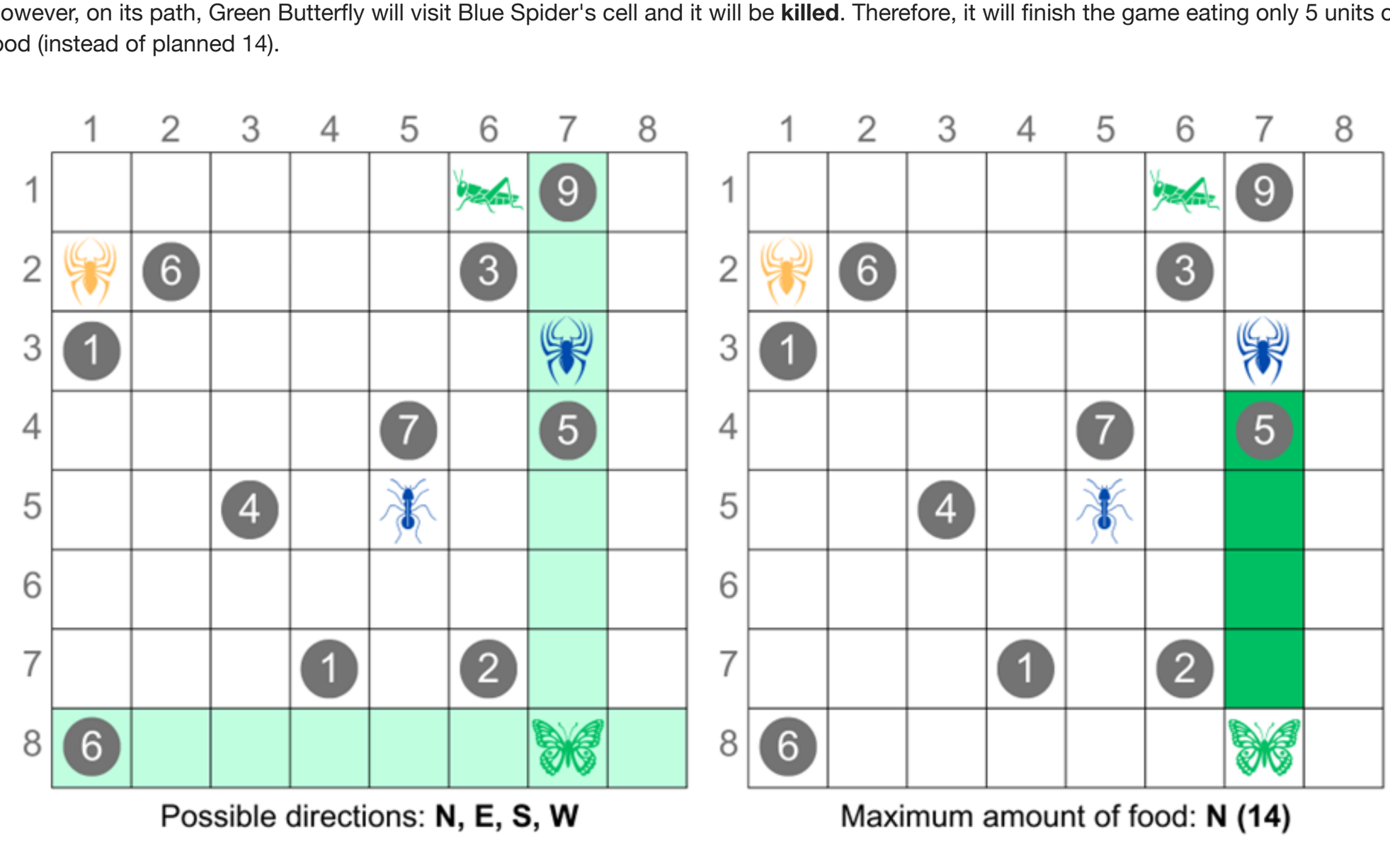


Figure 3. *Described starting board configuration*

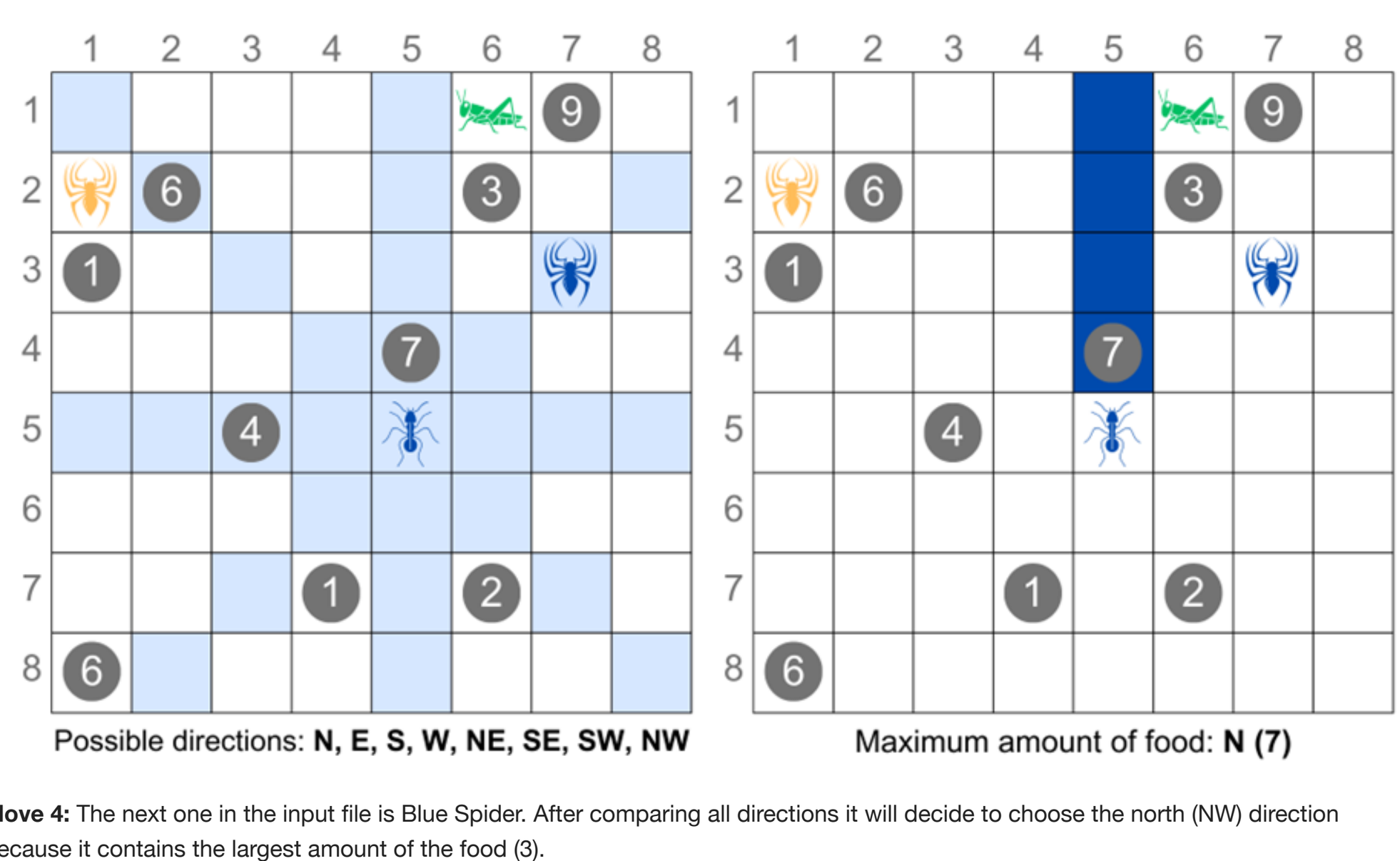
Move 1: Since Red Grasshopper is the first in the input, it will move first. After checking all possible directions, it will decide to choose north (N) because it contains the largest amount of food (5). Then, it will move north till the end of the board.



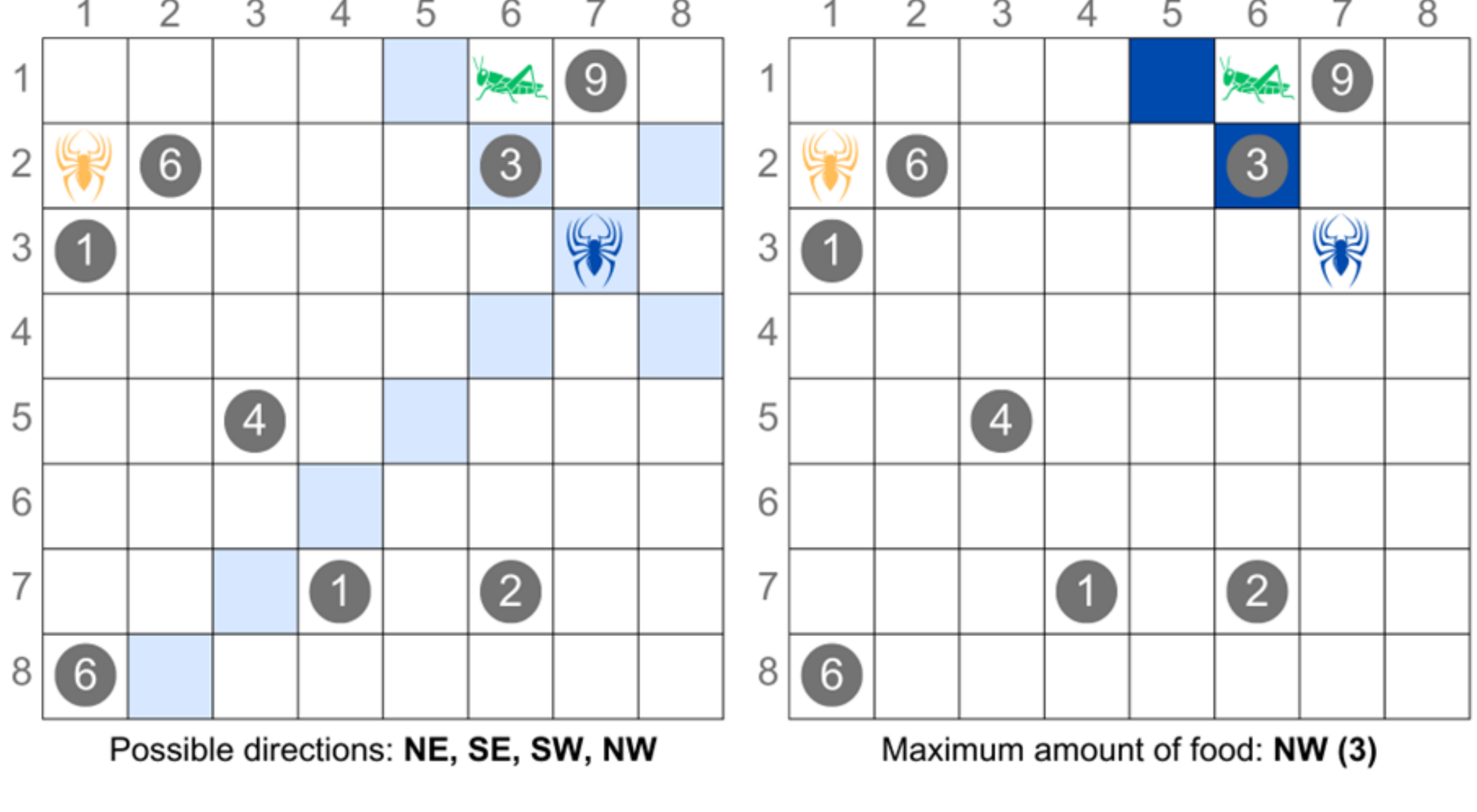
Move 2: The next one in the input file is Green Butterfly. After comparing all directions it will decide to choose north (N) direction because it contains the largest amount of the food (14). Note that the configuration of the board was changed after Red Grasshopper left it. However, on its path, Green Butterfly will visit Blue Spider's cell and it will be **killed**. Therefore, it will finish the game eating only 5 units of food (instead of planned 14).



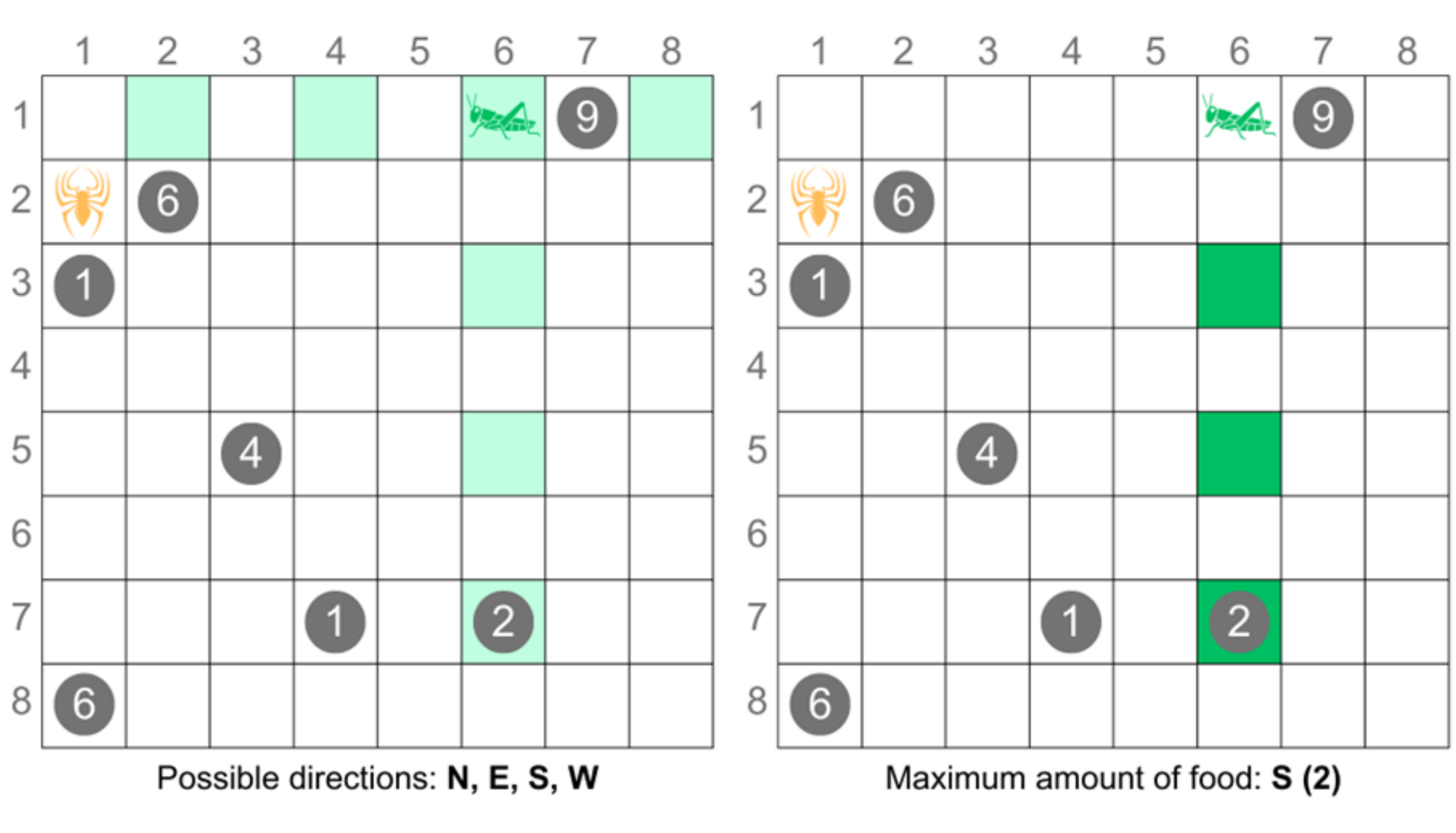
Move 3: The next one in the input file is Blue Ant. After comparing all directions it will decide to choose north (N) direction because it contains the largest amount of the food (7).



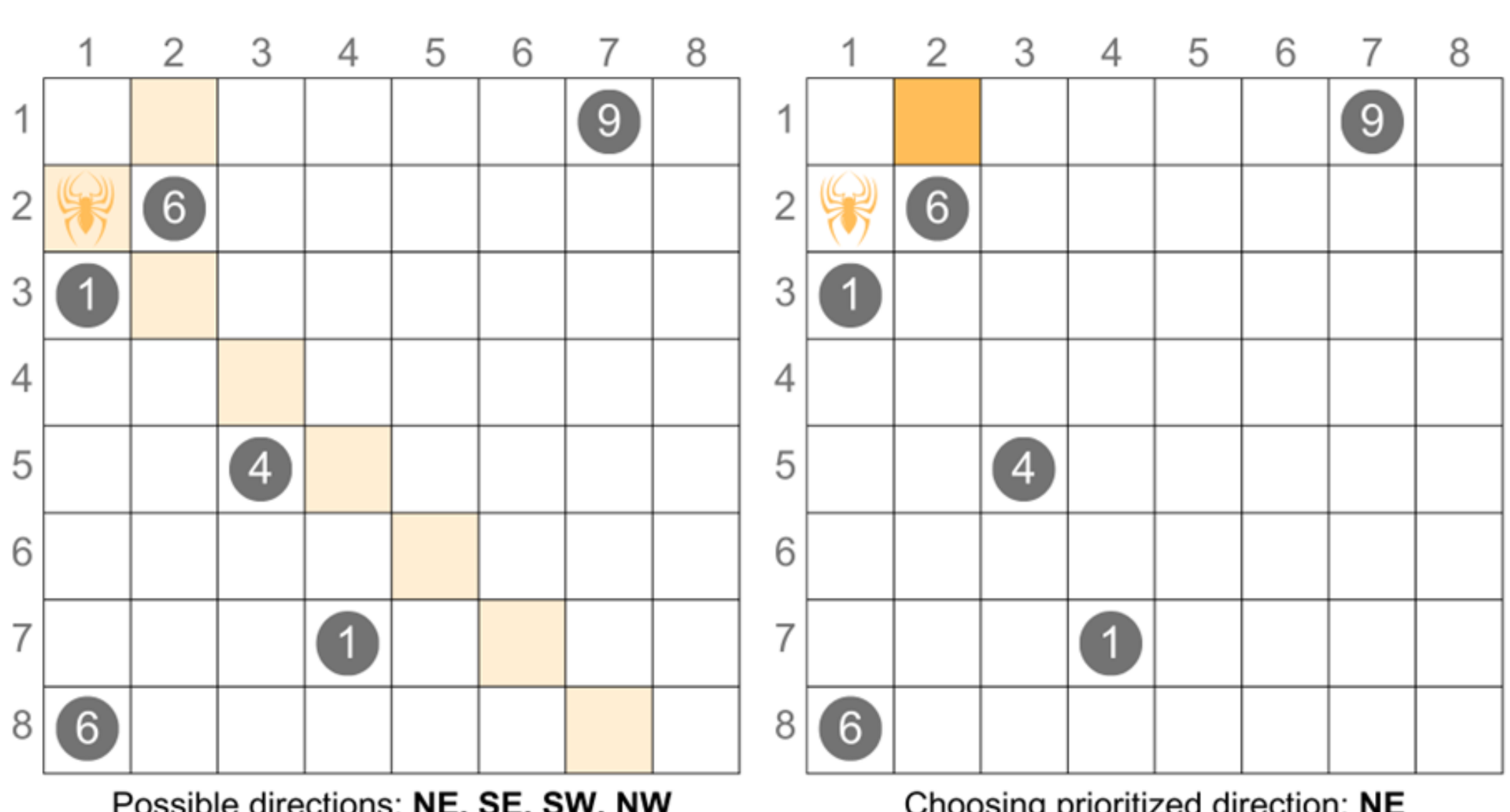
Move 4: The next one in the input file is Blue Spider. After comparing all directions it will decide to choose the north (NW) direction because it contains the largest amount of the food (3).



Move 5: The next one in the input file is Green Grasshopper. After comparing all directions it will decide to choose the south (S) direction because it contains the largest amount of the food (2).



Move 6: The last one in the input file is Yellow Spider. After checking all directions it will understand that there are no food points on any path. Therefore, it will use the prioritization of the directions provided above and choose north-east (NE).



Finally, the result of this game should be as follows:

- Red Grasshopper North 5
- Green Butterfly North 5
- Blue Ant North 7
- Blue Spider North-West 3
- Green Grasshopper South 2
- Yellow Spider North-East 0

Input

The input file (**input.txt**) should contain the following lines:

- The first line of the input should contain an integer ***D*** ($4 \le D \le 1000$), which represents the size of the board (the board is $D \times D$).
- The second line of the input should contain an integer ***N*** ($1 \le N \le 16$), which represents the number of insects of the board.
- The third line of the input should contain an integer ***M*** ($1 \le M \le 200$), which represents the number of *food points* on the board.
- The following *N* lines of of the input should contain four values separated by a single space in the following format: `Color InsectType XCoordinate XCoordinate`
- The following *M* lines of of the input should contain three values separated by space in the following format: `FoodAmount XCoordinate YCoordinate`

Note that coordinates are indexed from 1 to *D*.
Note that there will always be a **new line character** at the end of the input file.

Output

First of all, you need to check the input data for potential violations of the above mentioned rules. Here is the list of error messages that you should print to your output file (**output.txt**) in case of any errors in the input file:

- `Invalid board size` - should be printed if the board size *D* is out of the boundaries.
- `Invalid number of insects` - should be printed if the number of insects *N* is out of the boundaries.
- `Invalid number of food points` - should be printed if the number of food points *M* is out of the boundaries.
- `Invalid insect color` - should be printed if color of the insect is different from *Red*, *Green*, *Blue*, and *Yellow*.
- `Invalid insect type` - should be printed if the type of the insect is different from *Ant*, *Butterfly*, *Spider*, and *Grasshopper*.
- `Invalid entity position` - should be printed if the insect or food point is located out of the board.
- `Duplicate insects` - should be printed if there are more than one insect of the same color and type on the board, e.g., two blue ants on the board
- `Two entities in the same position` - should be printed in case of having more than one type of entity in the same cell.

You should always print only the first error found in the input file and then terminate the program without throwing any other exceptions. It is **guaranteed** that there will be **no other type of errors** in the input file.
For invalid inputs you should use **user-defined exceptions** to handle errors and report their messages using the overridden `getMessage()` method.

If there is no any of the above mentioned issues in the input file, your output file (**output.txt**) should contain *N* lines (one for each insect from the input), in the following format:

`Color InsectType Direction AmountOfFoodEaten`

Here `Direction` represents the direction that the insect will choose (in format *North*, *East*, *South*, *West*, *North-East*, *South-East*, *South-West*, *Norht-West*) while `AmountOfFoodEaten` represents the amount of food that the insect will eat on its path before leaving the board or being killed by the other insect.

Examples

input Скопировать

```
8
6
11
Red Grasshopper 5 4
Green Butterfly 8 7
Blue Ant 5 5
Blue Spider 3 7
Green Grasshopper 1 6
Yellow Spider 2 1
5 1 4
9 1 7
6 2 2
3 2 6
1 3 1
7 4 5
5 4 7
4 5 3
1 7 4
2 7 6
6 8 1
```

output Скопировать

```
Red Grasshopper North 5
Green Butterfly North 5
Blue Ant North 7
Blue Spider North-West 3
Green Grasshopper South 2
Yellow Spider North-East 0
```

input Скопировать

```
4
2
2
Red Grasshopper 3 2
Green Spider 3 3
100 3 4
50 1 2
```

output Скопировать

```
Red Grasshopper East 100
Green Spider North-East 0
```

input Скопировать

```
5
4
6
Red Ant 1 1
Red Spider 4 4
Red Butterfly 4 3
Green Butterfly 5 4
3 1 4
10 2 2
9 3 3
1 4 2
1 5 3
7 5 5
```

output Скопировать

```
Red Ant South-East 26
Red Spider South-West 1
Red Butterfly West 1
Green Butterfly North 3
```

input Скопировать

```
5
4
3
Orange Ant 2 1
Red Spider 4 4
Red Butterfly 4 3
Green Butterfly 5 4
3 1 4
10 2 2
9 3 3
```

output Скопировать

```
Invalid insect color
```

input Скопировать

```
5
2
3
Red Ant 2 1
Red Bug 4 3
5 1 4
6 2 3
9 3 3
```

output Скопировать

```
Invalid insect type
```

Note

Note that the **order of insects** in the output file should be the same as the order of the insects in the input file.

Note that there should be a **new line character** at the end of the output file.

Note that use of @SuppressWarnings for Checkstyle plugin will be considered as a **cheating case**.

→ Языки

Только перечисленные языки могут быть использованы для решения задач соревнования
Assignment 4:
- Java 11.0.6
- Java 17 64bit
- Java 21 64bit
- Java 1.8.0_241

→ Отослать?

Язык: Java 1.8.0_241
Выберите файл: Выбрать файл файл н...ыбран
Обратите внимание: в этой задаче используется файловый ввод/вывод
Отослать

→ Набранные баллы

	Баллы
A	