

### 3 Homework Assignment - Ape Intelligence

#### General information

Only those who passed the challenge program Entry Test can participate in the Challenge Program.

Your submission for all the parts of the exercise should be in Java. All the challenge exercises must be done individually.

#### Assignment description

The Earth is ruled by apes. Humanity is submitted and suffer. However, there is a hope. Uprising is underway. We need to know what the enemy plans are. The ape's military command encodes its orders to the troops. Fortunately for us, the ape military intelligence is not very advanced. We deciphered their code.

Not wanting to waste time interpreting the ape military commands, it was decided we make a program that automatically interprets commands and transforms them to English. You have been chosen to perform this task. Do it well, or we might end up getting wiped out...

#### Exercise

This exercise will be all about Java methods. Your job is to identify parts of the message and implement these in methods, and calling suitable other methods so that all these parts of the order are only implemented once. For example, on multiple occasions you will have to interpret an integer as a quantity, which means you should have a single method for that job, and call it. If you have multiple pieces of code doing the same thing, you will lose points, so take extra care in this respect. However, the goal of this exercise is not to remove any and all code duplication, so don't start a crusade to remove all of this: just ensure that parts of the language that are the same are implemented only once, and try to cluster parts of the language that logically belong together into methods as well.

Write a program that reads an ape's message and outputs it in English text. The message can have **3 different formats**, so, your program should have 3 parts.

#### Part 1: Apes attack!

The enemy have three kinds of commands they may give in battle: a command to attack with some troops ("resources"), to search for enemies (that's some of us!) or to retreat. The first integer of the message represents which command they give: a **0** means **attacking** with resources, a **1** means **searching** for humans and a **2** means an **order to retreat** to a certain place.

A **resource request** looks like "Attack with all your mounted orangutans." A resource request consists of two more integers: the first constitutes a quantity and the second the type of resources requested. If the quantity is 0, this means they want everything they have of that type.

The types of resources are:

- 1 "orangutans",
- 2 "chimps",
- 3 "gorillas",
- 4 "mounted chimps",
- 5 "mounted orangutans".

A **search request** consists of one mandatory part and one optional part. The mandatory part looks like "Search West" or " Search 42 caves" and the optional part looks like " and look for male humans" and is inserted right before the end of a message. The mandatory part either starts with a 0 or with a number greater than 0. If the number is 0, it is followed by a quantity and a type of a place or a direction to search.

If the starting number is a 0, it is a place, which are:

- 0 "hills",
- 1 "marshes",
- 2 "caves",
- 3 "woods"

If the starting number is greater than 0, it is a direction (without "the"!), which are:

- 1 "North",
- 2 "East",
- 3 "West",
- 4 "South",
- 5 "South-East",
- 6 "South-West",
- 7 "North-East",
- 8 "North-West"

The mandatory part may optionally be followed by an integer, which indicates searched objects. They are us:

- 0 "humans",
- 1 "human males",
- 2 "human females",
- 3 "human children",
- 4 "mutated humans"

A **retreat request** consists of one mandatory part and one optional part. The mandatory part looks like "Retreat to Forbidden Zone.". The optional part looks like " and move 23 mounted chimps to South-West " and is inserted right before the end of the message. The mandatory part consists of an integer which is the intended place to retreat:

- 0 "Ape City",
- 1 "Forbidden Zone",
- 2 "Rocky Mountains"

### Part 2: Ape war logistics

The ape military command also gives orders to send different supplies to their troops. They make items such as steel helmets or wooden shields. They love to make containers putting one item inside another. This kind of order will specify what kind of a container they order to make and to what direction send it.

An item has a **type**:

- 0 "wooden",
- 1 "steel",
- 2 "stone",
- 3 "cotton"

An **item** can be:

- 0 "cage",
- 1 "net",
- 2 "helmet",
- 3 "shield"

A container specification starts with a type followed by an item name. It is then followed either by a 4, in which case the container contains nothing, or followed by a 5, in which case two more container specifications follow this 5, or by a container specification which is contained in the container.

For example, "0 0 4" is a container (a wooden cage),

and "0 0 5 1 1 4 2 2 4" is a container (a wooden cage containing a steel net and a stone helmet).

A request starts with a container specification, and is optionally followed by a destination. A destination is optionally followed by a destination. Hence, the request "0 0 4 1 2 -1" would translate to "Pack a wooden cage and send one to North and send one to East".

### Part 3. Ape fortifications

The apes also give orders to build fortifications. They build them as towers according to the specifications they send as orders. They want their towers to consist of 1 by 1 by 1-meter boxes and 2 by 2 by 2-meter boxes, stacked on top of each other so that the 1x1x1 boxes are at the top and the 2x2x2 boxes are at the bottom. These are then placed on a concrete base. For some reason (perhaps to scare us humans), they want the 1x1x1 boxes to have images of their troops painted on them, while the 2x2x2 boxes are not allowed to be painted.

A box specification is either just the number 4, which means that the top of the tower is reached (and is therefore not really a box at all), or the type of the item (see Part 2), followed by the box that is on top of this box, optionally followed by a resource name (see Part 1). If the box is followed by a resource name, it is a 1x1x1 box, and otherwise (except of course if the box is just the integer 4) it is a 2x2x2 box.

Upon a 2x2x2 box, either a 2x2x2 box or a 1x1x1 box may be placed, but upon a 1x1x1 box only a 1x1x1 will be placed. So, the request "0 1 2 3 4 1 2 -1" should be interpreted as:

"Build a concrete base on top of which is a 2x2x2 wooden cube on top of which is a 2x2x2 steel cube on top of which is a 1x1x1 stone cube on top of which is a 1x1x1 cotton cube with chimps painted on the sides with gorillas painted on the sides."

### Input

The first line of any input will contain either "Part 1", "Part 2" or "Part 3", which tells your program which part of the exercise the input is for. The **next line** should contain an encrypted message which should consist of a sequence of integers. An integer value of -1 signifies the **end of the input**.

### Output

The output should be one line containing a decrypted message without a full stop.

Some sequences may result in orders that are nonsense (just as English has sentences which are syntactically correct but make no sense at all, such as "colorless blue concepts snore intelligently".) Your program should be able to handle them regardless. However, **if any part of a message is not syntactically correct (i.e., cannot be decrypted), your program has to output "..."** in the position of an error. Meanwhile, because the message can still be valuable for humanity, if it is possible, your program has to continue decryption.

### Examples

Input	Output
Part 1 0 0 4 -1	Attack with all your mounted chimps
Part 1 1 0 42 2 2 -1	Search 42 caves and look for human females
Part 1 1 2 -1	Search East
Part 1 4 1 -1	...

Part 1 2 2 23 3 5 -1	Retreat to Rocky Mountains and move 23 gorillas to South-East
Part 2 0 2 4 -1	Pack a wooden helmet
Part 2 2 0 1 3 5 0 2 4 3 1 4 1 3 -1	Pack a stone cage containing a steel shield containing a wooden helmet and a cotton net and send one to North and send one to West
Part 2 2 0 1 3 5 7 2 4 3 1 4 1 12 -1	Pack a stone cage containing a steel shield containing a ... helmet and a cotton net and send one to North and send one to ...
Part 3 0 1 2 3 4 1 2 -1	Build a concrete base on top of which is a 2x2x2 wooden cube on top of which is a 2x2x2 steel cube on top of which is a 1x1x1 stone cube on top of which is a 1x1x1 cotton cube with orangutans painted on the sides with chimps painted on the sides
Part 3 4 -1	Build a concrete base
Part 3 0 1 2 3 4 7 9 2 -1	Build a concrete base on top of which is a 2x2x2 wooden cube on top of which is a 1x1x1 steel cube on top of which is a 1x1x1 stone cube on top of which is a 1x1x1 cotton cube with ... painted on the sides with ... painted on the sides with chimps painted on the sides

### Specifications

1. Each encrypted message is a sequence of integers between -1 and 10000 separated by one space; you don't need to check this.
2. As your program is going to be automatically checked by Momotor, please, avoid in the output any (except an empty line in the end) extra strings or any extra characters, including leading or trailing spaces.
3. Your submission for all three parts is to be a single Java file.

### Restrictions and recommendations

You **are not allowed to use loop statements such as 'while' and 'for'**, as there is no need for them.

If you have implemented Part 1, you will hopefully have partitioned the program into several functions that each handle some part of the language.

If you do the exact same thing for the parts 2 and 3, this same approach will work without problems (unless your code has bugs): you can simply let a function call itself at the proper moment, which should be quite clear from the language definition. However, **using recursion to implicitly implement a loop is not allowed either.**