Favor for the Ringmaster
Systems Programming

1 Introduction

One morning, you wake up and find yourself in North Oxford in 1940, right next to J.R.R. Tolkien's house, serving as his assistant.

J.R.R. Tolkien is writing his novel, 'The Lord of the Rings'. Since it's a bit lengthy and challenging to manage, he asks one of his favorite assistants —you— to help him keep track of the characters' inventories.

Your duty is to invent something for Tolkien that keeps track of characters' inventories. As a time traveler familiar with the wonders of the C programming language, you decide to write an interpreter using C for Tolkien.

Figure 1: Tolkien requests a favor from you.

2 Details

Examining our C program, it remains ready to receive inputs until the user enters the "exit" command. This tool handles three input types:

- Sentence: Providing details about characters and their inventories. When provided with a valid sentence, the program outputs "OK"; otherwise, it prints "INVALID". However, the program should continue to run.
- Question: Prompting the program to find information for J.R.R. Tolkien's writing. The program accurately provides answers to questions.

- Exit Command: The program terminates.

2.1 Sentences

Our constructed language sentences are formed using these types:

2.1.1 Entities

- Subject(s): Describes entities or things of interest. Each subject is a single word that can contain uppercase and lowercase letters, including underscores. Multiple subjects can be detailed, separated by ‘and’. For example:
  - Gandalf
  - Frodo Baggins (INVALID: Contains space)
  - Frodo_Baggins
  - Aragorn and Sauron
  - Sam and Pippin and Mary
  - Gandalf1907 (INVALID: Contains digits)
- Item(s): Refers to objects associated with a non-negative integer quantity. Each item is a single word that can contain uppercase and lowercase letters, including underscores. Multiple items can be specified, each accompanied by a non-negative integer, and separated by ‘and’. The term does not get a plural suffix ‘s’ when it is specified as plural. For example:
  - Narsil (INVALID: Missing quantity)
  - 2 Narsil
  - 1 Palantir and 0 elven_rope and 44 Ormal
  - 1 Palantir and elven_rope and 44 Ormal (INVALID: Missing quantity)
- Location: Represents a place or setting using a single word that can contain uppercase and lowercase letters, including underscores. Describing multiple locations is not allowed. For example:
  - Gondor
  - Isendar and Rivendell (INVALID: Multiple locations)
  - Minas_Tirith

These entity names must not contain any special keywords defined in the language.

Keywords: {sell, buy, go, to, from, and, at, has, if, less, more, than, exit, where, total, who, NOBODY, NOTHING, NOWHERE}

In test cases, each name will belong to only one type of entity. For example, in a given scenario, 'Frodo' won't be assigned as both a subject name and an item name.

If the given entities do not follow these rules, the constructed sentence must be INVALID, and the program should print INVALID.

2.1.2 Actions

- "buy": Subject(s) acquire item(s) from an infinite source. After this action, the quantity of purchased item(s) in the inventory is increased. For example:

  Subject(s) buy Item(s)

  - Gandalf buy 5 bread
  - Gandalf and Gollum buy 2 bread
  - Gandalf and Gollum buy 6 bread and 4 water

- "buy ... from": Subject(s) acquire item(s) from another subject, which is a finite source. If there are multiple buyers, the seller provides the given amount of items to each buyer. The buy operation is done only if the seller has enough items for all buyers. Otherwise, do nothing. In the buy action, there cannot be more than one seller subject. The seller and buyer cannot be the same person. After this action, the quantity of purchased item(s) in the inventory for buyers is increased, while for the seller, it is decreased. For example:

  Subject(s) buy Item(s) from Subject

  - Gandalf buy 5 bread from Aragorn
  - Gandalf and Gollum buy 2 bread from Gollum (INVALID: Subject serving as both the seller and the buyer)
  - Gandalf and Gollum buy 6 bread and 4 water from Aragorn
  - Gandalf and Gollum buy 6 bread from Aragorn and Legolas (INVALID: Multiple sellers)

- "sell": Subject(s) give item(s) to an infinite source. After this action, the quantity of sold item(s) in the inventory is decreased. If the subject(s) do not have enough item(s), do nothing. For example:

  Subject(s) sell Item(s)

  - Gandalf sell 5 bread
  - Gandalf and Gollum sell 2 bread
  - Gandalf and Gollum sell 6 bread and 4 water

- "sell ... to": Subject(s) which are a finite source, provide item(s) to another subject. If there are multiple sellers, each provides the given amount of items to the buyer. The sell operation is done only if all the sellers have enough items for the buyer. Otherwise, do nothing. In the sell action, there cannot be more than one buyer subject. The seller and buyer cannot be the same person. After this action, the quantity of sold item(s) in the inventory for sellers is decreased, while for the buyer, it is increased. For example:

  Subject(s) sell Item(s) to Subject

  - Gandalf sell 5 bread to Aragorn
  - Gandalf and Gollum sell 2 bread to Gollum (INVALID: Subject serving as both the seller and the buyer)
  - Gandalf and Gollum sell 6 bread and 4 water to Aragorn
  - Gandalf and Gollum sell 6 bread to Aragorn and Legolas (INVALID: Multiple buyers)

- "go to": Subject(s) move to a location. It is possible for subject(s) to move to their current location. Subject(s) cannot go to multiple locations at the same time. Remember, describing multiple locations is not allowed. For example:

  Subject(s) go to Location

  - Gandalf go to Shire
  - Frodo and Sam go to Mordor
  - Frodo go to Mordor and Isengard (INVALID: Multiple locations)

It is possible to create action series called Action(s). It starts with a single action, followed by an 'and' and additional actions, optionally. Each action in an action series is performed from left to right separately.

Action(s): Frodo and Sam go to Mordor and Gandalf sell 5 bread and 3 sword to Aragorn and Legolas buy 3 hairclip

At the beginning, each subject has 0 items in their inventory and is located in NOWHERE.

2.1.3 Conditions

- "at": Specifies the location where the subject(s) are present. For this condition to be true, all subjects mentioned should be at the specified location. Remember, describing multiple locations is not allowed. For example:

  Subject(s) at Location

  - Gandalf at Shire
  - Frodo and Sam at Mordor
  - Frodo at Mordor and Isengard (INVALID: Multiple locations)

- "has": Indicates possession, stating that the subject has a certain quantity of an item. For this condition to be true, each subject mentioned should have the specified amount of the items exactly. For example:

  Subject(s) has Item(s)

  - Gandalf has 5 bread
  - Gandalf and Gollum has 2 bread
  - Gandalf and Gollum has 6 bread and 4 water
  - Gandalf has bread (INVALID: Missing quantity)

- "has less than": Describes a condition where the subject possesses an amount of an item that is less than a specified value. For this condition to be true, each subject mentioned should have less than the specified amount of the items. For example:

  Subject(s) has less than Item(s)

  - Gandalf has less than 10 ring
  - Gandalf and Gollum has less than 5 sword and 3 map
  - Gandalf has less than (INVALID: Missing items)

- "has more than": Describes a condition where the subject possesses an amount of an item that is greater than a specified value. For this condition to be true, each subject mentioned should have more than the specified amount of the items. For example:

  Subject(s) has more than Item(s)

  - Gandalf has more than 10 ring
  - Gandalf and Gollum has more than 5 sword and 3 map

  - Gandalf has more than ring (INVALID: Missing quantity)

These condition types are exclusively used within if statements and cannot exist independently.

It is possible to create condition series called Condition(s). It starts with a single condition, followed by an ‘and’ and additional conditions, optionally. Each condition in a condition series is assessed from left to right separately. If all conditions are true, the overall condition is considered true. If any of the conditions is false, the overall condition is considered false.

Condition(s): Frodo at Mordor and Gandalf has lower than 10 ring and Aragorn and Legolas has 5 map

2.1.4 Constructing a Sentence

Sentences can be made in 3 different ways:

- Basic Sentences: Composed of a set of actions that the program will perform. For example :

  Actions(s)

  - Legolas and Gimli go to Rivendell and Legolas and Gimli buy 2 elixir and 1 map from Arwen
  - Aragorn and Frodo sell 3 dagger to Gimli and Pippin go to Shire
  - Saruman sell 4 staff to Frodo
  - Aragorn and Legolas go to Lothlorien and Aragorn and Legolas buy 2 lembas_bread and 1 bow from Galadriel and Gimli sell 3 axe

- Conditional Sentences: Include actions that will only be executed if specific conditions are met. If all given conditions are true, then all actions will occur from left to right. For example:

  Action(s) if Condition(s)

  - Gandalf sell 3 sword to Aragorn if Frodo has more than 5 ring and Legolas and Gimli at Rivendell
  - Frodo and Sam go to Mount_Doom if Gollum has 1 the_One_Ring and Gandalf has less than 3 staff and 2 bread
  - Frodo and Sam go to Mordor and Gandalf sell 5 bread if Frodo has more than 3 ring

- Sequential Sentences: Combine actions and conditions in a sequence, allowing for a series of linked instructions. Each sentence is executed separately, following a leftto-right order. For example:

  Sentence-1 and Sentence-2 and Sentence-3

  - Frodo and Sam go to Bree and Frodo buy 3 map from Aragorn and Sam sell 2 dagger to Legolas if Frodo has more than 2 ring and Legolas and Gimli at Bree and Frodo and Sam go to Rivendell if Aragorn has 5 map and Frodo has less than 5 potion and Sam has 3 dagger and Frodo sell 1 potion to Arwen and Legolas and Gimli go to Rivendell

2.2 Questions

- Quantity asking (total ... ?): It inquires about the total count of a specific item for the mentioned subjects. The question is restricted to a single item and cannot involve multiple items. The result integer should be returned in a single line as the answer. For example:

  Subject(s) total Item ?

  Example: Gandalf and Frodo total ring ? → 5

- Location asking (where ?): It inquires about the current location of a specified subject. The question will involve only one subject. For example:

  Subject where ?

  Example: Frodo where ? → Rivendell

- Presence in a location (Who at ... ?): It seeks information about the subjects present in a specified location. Remember, describing multiple locations is not allowed. The response should provide a list of all subjects located in the given place in a single line, separated by 'and'. The order of the answer does not matter; any order will be accepted. If there is no one at the specified location, the output should be "NOBODY". For example:

  Who at Location ?

  Example: Who at Rivendell ? → Frodo and Gimli

- Inventory inquiry (total ?): This question type aims to retrieve information about the complete inventory of the specified subject. The question is not limited to a specific item but encompasses all items present in the subject's inventory. The result should provide details about each item, including its name and quantity, in a single line as the answer, separated by ‘and’. There will be only one subject asked at each of this question type. The order of the answer does not matter; any order will be accepted. Printing items with a quantity of 0 in the inventory will not be accepted. If a subject doesn’t have any items in their inventory, the output should be "NOTHING". For example:

Subject total ?

Example: Gandalf total ? → 5 ring and 3 staff and 2 bread

2.3 Exit Command

When the "exit" command is entered, the program should terminate without encountering any errors.

3 Input & Output

- There will be no expressions or assignments with a result that exceeds a 32-bit number.
- All words are case-sensitive; For example: ‘Shire’ and ‘shire’ are not the same.
- Sentences or questions will consist of 1024 characters at most.
- Each input should be read after printing ">> " to the terminal.
- There might be multiple spaces between words; it’s valid for both input and output. There won’t be a test case with escape sequences like \r and \t. Only white-space and \n at the end of the line will be given in the input.
- If a sentence or question is INVALID, no actions should be executed based on the given input, even if it contains correct sentences.
- The execution of the program will be done through the interpreter screen working in the terminal - in other words, it won’t be a file-based program. It should work just like the Python interpreter.

-

  Remember, when provided with a valid sentence, the program outputs "OK"; otherwise, it prints "INVALID". However, the program should continue to run.

- An empty line will not be given as input.
- Test cases will not exceed a maximum of 128 lines. Focus on code correctness rather than optimization for this project.

3.1 Examples to Rule Them All

These examples address the mentioned details but do not cover all edge cases.

- Example 1 : General expectations from the program.

```
$ make
$ ./ringmaster
>> Frodo go to Rivendell
OK
>> who at Shire ?
NOBODY
 >> Frodo         where ?
Rivendell
>> Frodo sell 2 bread to Sam
OK
>> Sam total ?
NOTHING (Explanation: Frodo does not have enough bread to sell to Sam.)
>> Frodo and Sam buy 3 bread and 2 ring
OK
>> Sam sell 1 ring to Sauron
OK
>> Sauron where ?
NOWHERE
>> Sam total ?
3 bread and 1 ring
>> Frodo buy 2 palantir and Frodo go to NOWHERE
INVALID (Explanation: Frodo can not go to a place which is a keyword.) >> Frodo total
palantir ?
0
>> Frodo total Bread ?
0
```

```
>> Frodo go to mount_doom and Gandalf buy 3 arrow if Sauron has 1 ring and Frodo at Rivendell
OK
>> who at mount_doom ?
Frodo
>> Legolas buy 100 hairclip from Arwen if Galadriel at NOWHERE
INVALID (Explanation: Even though Galadriel is at NOWHERE, her initial location, NOWHERE is a
keyword and therefore cannot be specified in sentences or questions as entities. But keywords such
as NOWHERE, NOTHING, NOBODY can be at the program's output.)
>> Legolas total hairclip ?
0
>> Legolas buy 100 hairclip from Arwen if Gandalf has more than 2 arrow OK
>> Legolas total hairclip ?
0 (Explanation: Even though the previous sentence is correct, Arwen doesn't have enough hairclips to
give to Legolas.)
>> exit
```

• Example 2 : Basic transaction process with one or more items.

```
$ make
$ ./ringmaster
>> Galadriel and Elrond and Cirdan buy 100 Nenya and 100 Narya and 100 Vilya OK
>> Balrog and Saruman buy 10 Vilya and 10 Narya from Cirdan OK
>> Cirdan total ?
100 Nenya and 80 Vilya and 80 Narya >>
Balrog total ?
10 Narya and 10 Vilya
>> Balrog and Cirdan sell 10 Nenya to Legolas
OK
>> Legolas total Nenya ?
0 (Explanation: The previous statement is correct, but since Cirdan doesn't have any Nenya
rings, the action is canceled for both.)
>> Balrog and Cirdan sell 10 Vilya to Legolas
OK
>> Legolas total Vilya ?
20
>> Balrog and Cirdan total Narya ?
90
>> exit
```

•

Example 3 : Entity names, question words and keywords are case-sensitive.

```
>> someone go to somewhere and somebody buy 1 somethings and 2 SoMeTHiNG if
someone has less than 2 somethings and 1 nothinG and someone go to
home_of_someone if someone has 10 something OK
>> someone where ?
somewhere >>
somebody where ?
NOWHERE
>> who at home_of_someone ?
NOBODY
>> everyone and somebody and someone go to everywhere and this is invalid if everyone at
everywhere
INVALID
>> everyone where ?
NOWHERE
>> exit
```

• Example 4 : When there are not enough items, even if the other items are sufficient, the action is not carried out.

```
>> Galadriel and Elrond buy 3 bread and 2 map
OK
>> Galadriel sell 3 bread and 3 map
OK
>> Galadriel total ?
3 bread and 2 map
>> Galadriel sell 3 bread and 3 map to Cirdan
OK
>> Galadriel total ?
3 bread and 2 map >>
Cirdan total ?
NOTHING
>> exit
```