CS1020 Take-home Lab #3

Exercise #3: Helicopter

(http://www.comp.nus.edu.sg/~cs1020/3_ca/labs.html)

Objective

The objective of this exercise is to understand and apply the concept of **circular linked list**.

Task Statement

An airport has **N** helicopter landing fields, numbered 0 … **N**-1. Prior to landing, a helicopter calls the airport control tower, reports its name and its destination landing field (let's call it landing field #m, m starts from 0). The control tower then checks if this landing field is available. If so, the tower directs the helicopter to the requested field for landing. Once a helicopter has landed, it occupies the landing field and that field is no longer available for others to land. The landing pad will only become available again when the helicopter takes off. If the destination landing field is occupied, the control tower will check for the next available landing field by scanning down the list of landing fields, starting from #m+1, until it spots an empty landing field and directs the helicopter there. If it reaches the end of the list, it will check from landing field number 0 again. If it comes back to the original destination landing field (#m) and still could not find an empty spot, the control tower will direct the helicopter to other airport.

Input

The first line consists of an integer **N** > 0 denoting the number of helicopter fields. The subsequent lines describe the helicopter incoming/outgoing events. Each incoming event is of the form: "Incoming *<name>* *<field#>* " where *<name>* is the name of the helicopter, *<field#>* is the destination landing field. Similarly, each outgoing event has the form: "Outgoing *<field#>*" where *<field#>* is the landing pad number that is supposed to become available again for landing. The end of input is encountered when reading a line containing just the word "END".

Note that short symbol such as **N** is used above for convenience. In your program, you are expected to give it a descriptive variable name.

Output

At the end of the operations, the system prints out a map of the landing fields (a series of lines each representing a field). An empty landing field is displayed as "Field *<field#>*: empty". An occupied field is displayed as "Field *<field#>*: occupied by helicopter: *<name>*".

Sample Input #1

```
5
Incoming GoldenBird1 4
Incoming GoldenBird2 0
Outgoing 4
Incoming GoldenBird3 1
END
```

Sample Output #1

```
Field 0: occupied by helicopter: GoldenBird2
Field 1: occupied by helicopter: GoldenBird3
Field 2: empty
Field 3: empty
Field 4: empty
```

Explanation

Incoming helicopter GoldenBird1 lands on field 4.
Incoming helicopter GoldenBird2 lands on field 0.
Helicopter (GoldenBird1) at field 4 leaves.
Incoming helicopter GoldenBird3 lands on field 1.

Sample Input #2

```
2
Incoming eagle1 1
Incoming eagle2 1
Incoming eagle3 0
Outgoing 1
Incoming eagle4 0
END
```

Sample Output #2

```
Field 0: occupied by helicopter: eagle2
Field 1: occupied by helicopter: eagle4
```

Explanation

Incoming helicopter eagle1 lands on field 1.
Incoming helicopter eagle2 requests for field 1, but it is occupied (by eagle1), so it lands on field 0.
Incoming helicopter eagle3 requests for field 0, but it is occupied, and all other fields are occupied as well, so it is directed to other airport.
Helicopter (eagle1) at field 1 leaves.
Incoming helicopter eagle4 requests for field 0, but it is occupied (by eagle2), so it lands on the next available field, which is field 1.