# Lab: Lists as Stacks and Queues

Problems for in-class lab for the [Python Advanced Course @SoftUni](https://softuni.bg/courses/python-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1830>

## Reverse Strings

Write program that:

* **Reads** an **input string**
* **Reverses** it **using a** stack
* **Prints** the result back at the terminal

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| I Love Python | nohtyP evoL I |
|  | seueuQ dna skcatS |

## Matching Brackets

We are given an arithmetic expression with brackets. Scan through the string and extract each sub-expression.

Print the result back at the terminal.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 + (2 - (2 + 3) \* 4 / (3 + 1)) \* 5 | (2 + 3)  (3 + 1)  (2 - (2 + 3) \* 4 / (3 + 1)) |
| (2 + 3) - (2 + 3) | (2 + 3)  (2 + 3) |

### Hints

* Scan through the expression searching for brackets
  + If you find an **opening** **bracket**, **push** the index into the stack
  + If you find a **closing** **bracket** **pop** the topmost element from the stack. This is the index of the opening bracket.
  + Use the current and the popped index to **extract** the sub-expression

## Supermarket

Write a program that **reads** an **input** consisting of a **name** and **adds** it to a **queue** until "**End**" is received. If you receive "Paid", **print** every customer name and empty the queue, otherwise we receive a client and we have to add him to the queue. When we receive **"**End**"** we have to print the count of the remaining people in the queue in the format: "{count} people remaining.**"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| George  Peter  William  Paid  Michael  Oscar  Olivia  Linda  End | George  Peter  William  4 people remaining. |
| Anna  Emma  Alexander  End | 3 people remaining. |

## Water Dispenser

Write a program that reads on the first line the starting **quantity** of water in a dispenser. Then on the next few lines you will be given the **names** of some people that want to **get water** (each on separate line) until you receive the command **"**Start**"**. Add those people in a **queue**. Finally, you will receive some commands until the command **"**End**"**:

* **{liters}** - Litters that the current person in the **queue** wants to get. Check if there is **enough** water in the dispenser for that person.
  + If there is enough water, print **"{person\_name} got water"** and remove him/her from the queue.
  + Otherwise, print **"{person} must wait"** and **remove the person** from the queue **without reducing** the water in the dispenser
* **refill {liters}** - add the given litters in the dispenser.

At the end print how many litters of water are left in the format: **"{left\_liters} liters left"**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 2  Peter  Amy  Start  2  refill 1  1  End | Peter got water  Amy got water  0 liters left | We create a queue with Peter and Amy. After the start command we see that Peter wants 2 liters of water (and he gets them). Water dispenser is left with 0 liters. After refulling, there is 1 liter in the dispenser. So when Amy wants 1 liter, she gets it and there are 0 liters left in the dispenser |
| 10  Peter  George  Amy  Alice  Start  2  3  3  3  End | Peter got water  George got water  Amy got water  Alice must wait  2 liters left |  |

## Hot Potato

Hot potato is a game in which children form a circle and start passing a hot potato. The counting starts with the first kid. **Every nth toss the child left with the potato leaves the game**. When a kid leaves the game, it passes the potato along. This continues **until there is only one kid left**.

Create a program that simulates the game of Hot Potato. **Print** everykidthat is **removed** fromthe **circle**. In the end, **print** the kidthat is left **last**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Tracy Emily Daniel  2 | Removed Emily  Removed Tracy  Last is Daniel |
| George Peter Michael William Thomas  100000000 | Removed Thomas  Removed Peter  Removed Michael  Removed George  Last is William |
| George Peter Michael William Thomas  1 | Removed George  Removed Peter  Removed Michael  Removed William  Last is Thomas |