Tutorial series N°4

Exercise 1 :

Given a stack **`S`** of integers, the following questions are independent:

1. Write the procedure **`nbEvenOdd(S, nbEven, nbOdd)`** that returns the number of even and odd elements in a stack **`S`** passed as parameters. At the end, stack **`S`** should be empty.
2. Write the procedure **`reorganize(S)`** that reorganizes the stack **`S`** passed as a parameter so that negative elements are at the bottom and positive elements are at the top. Note: It is not required to sort the stack.
3. Write a function named **`reverse_stack(S)`** that takes a stack **`S`** as a parameter and returns a new stack containing the elements of **`S`** in reverse order. The original stack **`S`** should remain unchanged. Indication: Use two stacks.

Exercise 2 :

An arithmetic expression can contain digits, symbols, mathematical operators, and parentheses. Parentheses can be nested within each other, for example:

$$f(x) = \left(\left(x^2 + 2x + 3\right)/\left(\left(4x + 2\right)*3\right)\right)$$

Write a function that takes an arithmetic expression (in the form of a string) as a parameter and determines whether it is correctly parenthesized, meaning all parentheses are properly closed.

Indication: The expected solution involves using a stack.

Exercise 3:

A palindrome is a string of characters that reads the same from left to right or from right to left (e.g., "level", "radar", "civic").

Using only one stack and one queue, write a function that tests whether a string (stored in a linked list of characters) is a palindrome or not.

Exercise 4 :

An equipment rental company receives orders every day. Each order has a number and a status (processed or not). At the end of each day, the responsible agent needs to enter all orders (processed and unprocessed) in the order they were received and store them in a data structure.

1. What is the most suitable data structure for this problem? Provide the declaration of this structure.
2. Write a procedure that takes all orders as input and returns the number of processed and unprocessed orders.
3. Write a function that reorganizes the data structure containing all orders by placing processed orders at the end in the order of their arrival.
4. Modify the previous function so that processed orders are placed at the end in reverse order of their arrival.

Additional Exercises

Exercise 5 :

Let **S** be a stack of integers.

1. Write the function **search(S, v)** that checks if the value **v** exists in the stack **S**.
2. Write the function **modify(S, v)** that doubles all values less than **v** only if **v** exists in the stack.

Exercise 6 :

Let **S** be a stack of integers.

1. Write the function **average(S)** that returns the average of the elements of a stack **S** passed as a parameter.
2. Write the procedure **binarize(S)** that sets to 0 all values less than the average of the stack and to 1 all values greater than or equal to the average.

Exercise 7:

Consider a stack of integers ordered in descending order.

Write a procedure that inserts a given value **v** into this stack if it does not exist.

Exercise 8:

1. Write the procedure **MinMax(S, min, max)** that returns the minimum and maximum values in a stack **S** passed as parameters.
2. Write the procedure **removeMinMax(S)** that removes the minimum and maximum values from the stack **S** passed as parameters and puts them into another stack.

Exercise 9 :

Same question as in Exercise 2, but here we consider that the formula can also contain brackets and braces, and these elements can be nested inside each other.

In this exercise, a formula is considered well-parenthesized when all parentheses, brackets, etc., are properly closed by a character of the same type, and when parentheses, brackets, and braces are correctly nested.

Exercise 10:

The following questions are independent:

1. Using only a queue, write the procedure **reverse** that reverses the elements of a stack **S** passed as parameters.
2. Write the procedure **split** that splits a queue of integers **Q** into two: one containing even elements and the other containing odd elements. At the end, the queue **Q** should be empty.
3. Write the function **isSorted** that checks if a queue **Q** passed as parameters is sorted.

Exercise 11:

In a cinema, spectators wait in line to buy their tickets. The line is represented by a queue of integers, where each integer represents a spectator. However, if a person has special needs, they must be prioritized and served first. A person with special needs is represented by the number 1.

Write a function named **add_spectator** that takes the queue and a person's number as parameters and adds this person to the end of the queue. If the person has special needs (represented by the number 1), he must be added to the front of the queue, ahead of all other people.

If multiple people with special needs are present in the queue, they should be processed one after the other, always in the order of arrival.