uc3m | Universidad Carlos III de Madrid

Data Structures and Algorithms.
Author: Isabel Segura Bedmar

Unit 2 – Linear ADT

Problem – Balanced arithmetic expression.

Implement a Python class to guess if the delimiters (,),{,},[,] in an arithmetic expression (e.j. [(5+x)-(y+z)]) are balanced.

- Correct expression example: ()(()){([()])}
- Incorrect expression example: ({[])}

Use a stack to implement your solution. Consider the following hints:

- If an opening symbol is found **[,(,{** it must be pushed.
- If a closing symbol is found **],},)** the element at the top of the stack must be queried. If both symbols belong to the same type, the element must be removed.
- The arithmetic expression is balanced if at the end of the process the stack is empty.

Problem 2 – Josephus problem.

In the Jewish revolt against Rome, Josephus and 39 of his mates were holding out against the Romans in a cave. With defeat imminent, they resolved that they would rather die than be slaves to the Romans. They decided to arrange themselves in a circle. One man was designated as number one, and they proceeded clockwise killing every seventh man (step). Josephus was among other things an accomplished mathematician; so he instantly figured out where he ought to sit in order to be the last to go. But when the time came, instead of killing himself he joined the Roman side.

Implement a method to find out what position Josephus should sit in order to not be killed. The solution should generalizate for any number of jewish soldiers and any step. The solution should use a queue of integers (each soldier is represented with a number from 1 to n).

In the following video, you can find a nice explanation of this problem.
https://www.youtube.com/watch?v=uCsD3ZGzMgE

Problem – Printer Queue

Implement a Python class, PrinterQueue, to manage a network printer. The printer can receive requests from different machines in a network. The requests should be printed by entry order. Each request includes the following information: id (String) of the machine that performs it (for example “I3493”) and the name of the file to print (for example “unit2.pdf”). Please, write a class, named Request, to represent a request.
The class, PrinterQueue, must implement the following operations:

- addRequest: takes a request as input and adds it to the set of requests.
- printWork: gets the first request and shows its data (id and name file) by console (it only simulates the imprension of the request) . The request has to be removed from the set of requests.
- getNumRequest(): returns the total number of requests.
- showAll(): shows all the requests that have have been not printed.
- printAll(): print all the requests. After processing a request, this must be removed.
- Include the needed instructions to test all the methods explained above.

Problem – Implementation of singly linked list using head.

Problem - Implementation of doubly linked list (with head and tail references).

Problem - Extending the doubly linked list class.

Given the DoublyLinkedList class (which you can download from aulaglobal).
Implement the following methods:

- getAtRev(self,index): which takes an index and returns the element at the index position starting from the end. For example, given the following list: A,B,C,D,E.
  - l.getAtRev(0) = 'E'
  - l.getAtRev(1)= 'D'
  - l.getAtRev(2)= 'C'
- getAtEff(self,index): as the getAt method, this new version also returns the element at the index position of the list. However, you must try to implement a more efficient method than the original method getAt, taking advantage of the doubly linked lists can be traversed in both forward and backward direction.
- Moreover, you must implement more efficient versions for the methods removeAt and insertAt.
- removeAll(self,e): which removes all the occurrences of e from the list. You must implement two different versions of the list:
  - a method that traverses all the nodes of the list for searching all the occurrences of e.
  - a method that exploits other methods such as removeAt.

Problem - Checking palindrome words by using a doubly linked list.

A palindrome word is one that reads the same backward as forward. Examples:

Anna, Level, Civic, Madam, Noon.

Implement a Python function that takes a word and returns true if it is palindrome, else false.

In your solution, you **have to use a doubly linked list** where each node contains only one character of the input word.