1. Créer une structure de liste chainée qui contient une seule information de type réel
2. Donner un algorithme qui permet d’ajouter un nouvel élément à la liste
3. Donner un algorithme qui permet de supprimer un élément à la liste
4. En supposant que la liste Tab est remplie, fournir un algorithme qui effectue la recherche d’un élément dans la liste:
   1. Dans le cas Tab non trié
   2. Dans le cas Tab trié
   3. Discuter de la complexité de chacun des deux cas précédents.
5. Fournir un algorithme qui effectue la suppression d’un élément de la liste :
   1. Dans le cas Tab non trié
   2. Dans le cas Tab trié
   3. Discuter de la complexité de chacun des deux cas précédents.

Links :

https://kingrayhan.medium.com/500-data-structures-and-algorithms-practice-problems-and-their-solutions-b45a83d803f0

https://www.techiedelight.com/linked-list-implementation-part-1/

**http://cslibrary.stanford.edu/105/**

**Problems**

1. **Check if an expression is balanced or not**

Given a string containing opening and closing braces, check if it represents a balanced expression or not.

For example,

{[{}{}]}[()], {{}{}}, []{}() are balanced expressions.

{()}[), {(}) are not balanced.

Hints

We can use a stack to solve this problem. The idea is to traverse the given expression, and

If the current character in the expression is an opening brace ( or { or [, push it into the stack.

If the current character in the expression is a closing brace ) or } or ], pop a character from the stack, and return false if the popped character is not the same as the current character, or it does not pair with the current character of the expression. Also, if the stack is empty, the total number of opening braces is less than the closing brace number at that point, so the expression cannot be balanced.

1. **Activity Selection Problem**

Given a set of activities, along with the starting and finishing time of each activity, find the maximum number of activities performed by a single person assuming that a person can only work on a single activity at a time.

For example,

**Input:** Following set of activities  
   
(1, 4), (3, 5), (0, 6), (5, 7), (3, 8), (5, 9), (6, 10), (8, 11), (8, 12), (2, 13), (12, 14)  
   
**Output:**  
   
(1, 4), (5, 7), (8, 11), (12, 14)

Hints

The activity selection problem is a problem concerning selecting non-conflicting activities to perform within a given time frame, given a set of activities each marked by a start and finish time. A classic application of this problem is scheduling a room for multiple competing events, each having its time requirements (start and end time).

Let’s assume there exist n activities each being represented by a start time si and finish time fj. Two activities i and j are said to be non-conflicting if si = fj or sj = fi.

We can solve this problem by being greedy. Using a [greedy approach](https://www.techiedelight.com/greedy-algorithm-problems/) will always result in an optimal solution to this problem. The idea is to initially sort the activities in increasing order of their finish times and create a set S to store the selected activities and initialize it with the first activity. Then from the second activity onward, include the activity in the activities list if the activity’s start time is greater or equal to the finish time of the last selected activity. Repeat this for each activity involved.