Faculty of MISM | 2023/2024
Department of Mathematics | Algorithms and data structures 2
First year LMD | Semester 2

Tutorial series N°3

Exercise 1 :

Consider the following memory state :

Variable	Address	Value
`x:integer`	2686748	
`z:integer`	?	
...	...	...
`y:integer`	2686760	
...	...	...
`p:^integer`	2686866	

1) Give the address of variable **z**, knowing that it is located just after variable **x** in memory.

2) What will be the values of the variables used after the execution of each of the following instructions:

`Var x,z,y:integer;` `p:^integer;`	**x**	**Z**	**y**	**p**
`x ← 5;`				
`z ← 8;`				
`p ← @x;`				
`y ← p^*p;`				
`y ← p^*x;`				
`y ← @p;`				
`y ← (@p^)^+2;`				
`(p+1)^ ← 3;`				
`p ← p+3;`				
`p^ ← (p-2)^*4;`				
`Allocate(p);`				

Exercise 2 :

A curve is a set of points in the plane. We define a polygon as a curve closed upon itself.

Therefore, the polygon consists of a set of points in the plane, such that the first point of the polygon coincides with the last point.

1) Using pointers, write a procedure to reserve the necessary space to store the coordinates of **n** points (where **n** is passed as a parameter), input the coordinates of these points, and display whether they form a polygon.
2) Test this procedure in a main algorithm.

Exercise 3 :

Using pointer formalism:

1) Write a C language function, **`fillArray`**, that reads the number of elements in an array of floats, allocates memory for the entire array accordingly, reads the elements of the array from the keyboard, and returns the address of the array. The function should also return the number of elements to the calling program in a variable passed by reference.
2) Write the main function, that fills an array by calling the previous function and displays its elements.

Exercise 4:

Let `L` be a linked list of integers. We want to determine if all elements in `L` are equal. To address this, a method involves finding the minimum and maximum values in `L`. If these two values are equal, it indicates that all elements share the same value. While not the optimal solution, it provides an approach.

1) Write a function to return the minimum value in a linked list `L` passed as a parameter.
2) Write a function to return the maximum value in a linked list `L` passed as a parameter.
3) Write a function that takes a list of integers as input and returns whether all elements are equal.

Exercise 5:

The objective of this exercise is to merge two linked lists, `L1` and `L2`, to create a new list composed of the elements of `L1` followed by those of `L2`. At the end, both `L1` and `L2` should be empty.

Several solutions exist; the simplest is to link the last element of `L1` with the first element of `L2`, and then point the head of the "result" list to the first element of `L1`.

Write a function to implement the described process.

Exercise 6:

Consider a list of characters, `L`:

1) Write a function that returns the address of the second-to-last element in a linked list `L` passed as a parameter.
2) Write a procedure that swaps the positions of the first node and the last node in a linked list `L` passed as a parameter.

Exercise 7:

Consider a linked list of integers, `L`:

1) Write the recursive procedure `display(L)` to display the elements of a linked list of integers passed as a parameter.
2) Write the recursive function `search(L, v)` that searches and returns whether the value `v` belongs to the list `L`.
3) Modify the previous function to return the address of the element containing the value (`Nil` if it does not belong to the list) and not just whether it exists or not.
4) Write the recursive procedure `sum(L, sEven, sOdd)` that calculates and returns the sum of even elements and the sum of odd elements in a linked list `L` passed as input.

Exercise 8:

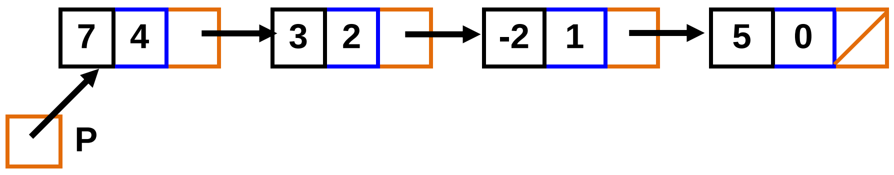
Consider a list of real numbers.

1) Write the function **`insertSorted(L, v)`** that inserts an element into a linked list sorted in ascending order, ensuring that the list remains sorted after insertion.
2) Write the main algorithm that prompts the user to enter 10 real numbers and inserts them into a linked list in such a way that the resulting list is sorted in ascending order.

Exercise 9:

Polynomials can be represented by a linked list, where each term of the polynomial is stored in a node of the list containing the degree (degree) of the term and the corresponding coefficient (coeff).

For example, the polynomial $7x^4 + 3x^2 - 2x + 5$ is represented as follows:



1) Provide the data structure **`Poly`** allowing to represent a polynomial.
2) Write the function **`evaluate(P, x)`** that evaluates a polynomial **`P`** for a given value **`x`**. The polynomial and the value **`x`** are passed as arguments.

Additional Exercises

Exercise 10:

Always using the pointer formalism, extend Exercise 3 by adding a procedure that determines and returns the maximum and minimum values of an array of real numbers of any size. Therefore, four parameters need to be considered: the array, its size, the maximum value, and the minimum value.

Exercise 11:

Consider the sequence u_n defined by:

$u_0 = 1$ and $u_{n+1} = 3u_n + 1$

Using pointer formalism:

1) Write a function in C that takes an integer **`n`** as a parameter and returns an array containing the first **`n`** terms of the sequence u_n.
2) Write the main program that inputs an integer **`n`** and displays the first **`n`** terms of the sequence u_n. These terms should be calculated and returned by the previous function.

Exercise 12:

Let **`L1`** and **`L2`** be two lists of integers.

Write a procedure **`DisplayCommon(L1, L2)`** that displays all elements common to both lists **`L1`** and **`L2`**.

Exercise 13:

1) Write the function `isSorted(L)` to determine if a linked list **`L`** passed as a parameter is sorted (in ascending order) or not. The function should return a boolean value.
2) Write the function `searchSorted(L, v)` that searches if a value **`v`** exists in a sorted linked list **`L`**. This function should perform the minimum number of tests possible.

Exercise 14:

Write the function `delete(L, v)` that removes the element containing the value **`v`** from a linked list **`L`** passed as a parameter. Note that the deletion could be at the beginning, at the end, or in the middle.

Exercise 15:

Write the function **`insertPosition`** that takes a list **`L`**, an integer **`v`**, and a position **`k`** as parameters. It inserts the value **`v`** at the **`k`**th position in the list **`L`** and returns the updated list.

This function returns a boolean value indicating whether the insertion was successful or not.

Exercise 16:

A company wants to create a program to manage its list of employees. Each employee is characterized by his last name, first name, national identification number, age, seniority, position, and salary. The program should allow the company to add new employees (hiring), remove employees (resignation, termination, retirement, etc.), modify the information of an employee (change in position, salary adjustment, correction of erroneous information, etc.). The program should also automatically increase the age and seniority of employees at the beginning of the year. After analyzing the company's needs, answer the following questions:

1) Propose a data structure that allows the company to organize and manage the information of its employees.
2) In your opinion, what are the most important functions to implement according to the needs specified by the company?

   Provide the implementation of these functions.