LO21-2011A-FS01-01

Final Exam – LO21 et LO27
2 hours
No document authorized

The scale for each exercise is provided for information purposes and may be amended during the correction

Exercise 1: Theater Management (10 points)

Note: In this exercise, you can use the functions defined during the lesson to manipulate the abstract data type List (is_empty, insert_head, insert_tail, etc.).

Consider a row in a theater as a list of places/seats. Each place is characterized by its number or rank (an integer between 1 and N, where N is unknown) and a state (free or busy). A row is sorted in ascending order of seat's numbers.

Question 1: (2 points)
Given a row, write the recursive algorithm of the subroutine **nbFreePlace** to calculate the number of free places.

Question 2: (3 points)
Given two rows, write the recursive algorithm of the subroutine **commonFreePlace** to calculate the number of free places at the same rank in the two rows.

Consider a theater as a linked list of numbered rows. Each element of this list is characterized by the row number (an integer) and the row is represented by a linked list of places. A place is defined as before (number of place and state).

Question 3: (2 points)
Give the C declaration of the types required for the declaration of a theater: *Place* type, *Row* type, *Theater* type.

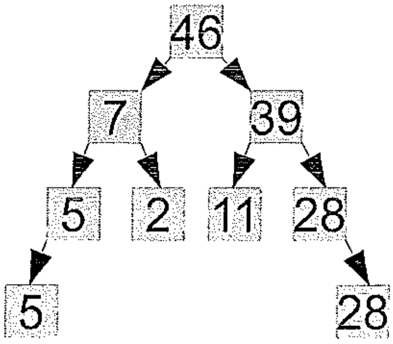
Question 4: (3 points)
Given a theater, write in C the iterative subroutine **getFreePlaces** to build a new list *FreeList* containing all the free places of the theater and their row number. Each element of the new list is a pair consisting of the row number and the number of the free place.
Give the C declaration of the type *FreeList*.

Exercise 2 : Binary Tree and Binary Search Tree (7 points)

Question 1 : Binary Tree SUM (3 points)

A SUM binary tree is defined such that for every non-leaf node N in this tree, the value of N is equal to the sum of the values of its right and left children. The value of a leaf node is fixed initially.
Provide the recursive algorithm of the function **isBTSum** that returns true if the specified binary tree is a SUM binary tree, false otherwise.

Example of SUM binary tree :

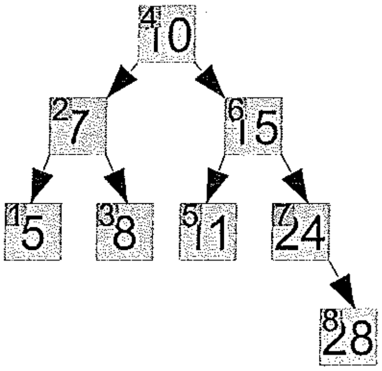


Question 2 : Access to the k^th element of a binary search tree (4 points)

Consider A a binary tree containing strictly positive integer values.
Assuming we have the function **size(a : Binary Tree) : Integer** that computes the number of nodes inside the tree A.
Provide the recursive algorithm of the function **findPos** enabling the access to the k^th element of the binary search tree a following the ascending order, if k≤0 or k>size(a) then findPos = -1

Example :
Let a the following tree,

- if k=-6, findPos = -1
- if k=1, findPos = 5
- if k=4, findPos = 10
- if k=8, findPos = 28
- if k=15, findPos = -1



Exercise 3 : Automata (3 points)

Build the automaton on the alphabet of the digits from 0 to 9 (A = [0 .. 9]) recognizing even numbers.