Data Structures and Algorithms

Computer Science Degree

Final Exam. June 27, 2017.

Name: _____ **Group:** _____

Laboratory: _____ **Desk:** _____ **DOMjudge User:** _____

1. (3 points) A **generic Fibonacci series** is a Fibonacci series where the first two terms of the series are two arbitrary integer numbers. For example, the following series is a generic Fibonacci sequence:

3 7 10 17 27 44 71 115

Specify, design and code an iterative algorithm that receives an array of integer numbers and calculates the length of the longest segment in which the elements form a generic Fibonacci sequence. Write the invariant and the termination function that prove the correctness of the coded algorithm. Finally, calculate and justify the complexity of the algorithm.

The function must receive the length and the array, and it will write, in separate lines, the length of the longest segment.

Entrada					**Salida**
n	v				
1	3				1
2	3	7			2
2	7	3			2
3	3	7	11		2
3	3	7	10		3
4	1	1	3	7	2
4	1	2	3	7	3
4	3	7	10	17	4
4	-1	6	5	10	3
4	5	3	1	2	2
4	5	3	-1	2	3

2. (2 points) We say an array of integer numbers is **shy** when its elements are sorted and the maximum number of times that an element appears in the array does not exceed its value. For example, the following arrays are shy

1 2 2 3 3 3 4 4 4 2 3 3 5 5 5 5 5

but these are not

0 1 2 2 3 3 3 4 4 1 1 2 2 3 3 3 7

We need to code an algorithm that, for a given array of length n, an initial value ini and a final value fin, being $ini < fin$, writes all the possible shy arrays of length n that contain the values between ini and fin. The arrays must be written in lexicographical order.

The program must read the values of n, ini and fin, and must write, in separate lines, all the possible shy arrays.

Entrada			**Salida**			
n	ini	fin				
4	2	4	2	2	3	3
			2	2	3	4
			2	2	4	4
			2	3	3	3
			2	3	3	4
			2	3	4	4
			2	4	4	4
			3	3	3	4
			3	3	4	4
			3	4	4	4
			4	4	4	4

3. (2 points) We say a node in a binary tree is *singular* if the sum of the values of its ancestors (including the root) equals the sum of the values of its descendants. Code an algorithm that, provided a binary tree of integer numbers, returns the number of singular nodes that it contains.

Apart from coding the algorithm, justify its complexity.

4. (3 points) We have been asked to implement a system to manage the admission of patients in the Emergency Service of a Hospital. When a patient arrives to the service, he provides his contact information and he is assigned a unique ID code (a non-negative number). After that, the patient waits to be seen in strict order of arrival. Once he has been seen, his data are erased from the system. At any moment, a patient can decide to leave the Emergency Service. In this case, there is an exit control where he is asked for his ID and all his data are erased from the system.

The system must be implemented using the ADT `AdmissionManagement` with the following operations:

- `create()`: constructor that creates an empty system.
- `addPatient(id, name, age, symptoms)`: it adds a new patient to the system, with identification code `id`, name `name`, age `age` and with a description of the symptoms `symptoms`. In case the code is duplicated, the operation shows an error *"Paciente duplicado"*.
- `patientInfo(id, name, age, symptoms)`: `name`, `age` and `symptoms` return the information associated to the patient with code `id`. In case the code does not exist, the operation shows an error *"Paciente inexistente"*.
- `next(code)`: it stores in `code` the code of the next patient to be seen. In case there are no more patients, the operation shows an error *"No hay pacientes"*.
- `morePatients()`: returns `true` if there are more patients waiting, and `false` otherwise.
- `erase(code)`: it erases from the system all the data corresponding to the patient with code `id`. If the patient does not exist, the operation has no effect.

Since this is a critical system, the implementation of the operations must be as efficient as possible. Therefore, you must choose an appropriate representation for the ADT, code the operations and justify the resulting complexity.