Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Final, 27 de Junio de 2017.

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

1. (3 puntos) Una **sucesión de Fibonacci genérica** es aquella en la que los dos primeros términos de la sucesión son dos números enteros cualesquiera. Por ejemplo, la siguiente sucesión cumple que es una sucesión de Fibonacci genérica:

3 7 10 17 27 44 71 115

Especifica, diseña e implementa un algoritmo iterativo que, dado un vector de números enteros, calcule la longitud del mayor segmento que cumpla que sus elementos forman una sucesión de Fibonacci genérica. Escribe el invariante y función de cota que permitan demostrar la corrección del algoritmo implementado. Por último, calcula y justifica el coste del algoritmo conseguido.

La función recibirá la longitud y el vector, y dará como salida, en líneas separadas, la longitud del segmento más largo.

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 puntos) Se dice que un vector de números enteros es **tímido** si sus elementos están ordenados y ningún elemento aparece más veces de lo que indica su valor. Por ejemplo, serían tímidos

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

pero no lo serían

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

Se desea implementar un algoritmo que, dada una longitud n, un valor inicial ini y un valor final fin, con $ini \leq fin$, escriba todos los posibles vectores tímidos de longitud n con valores comprendidos entre ini y fin. Los vectores se deberán escribir en orden lexicográfico.

El programa leerá de la entrada los valores de n, ini y fin y escribirá, en líneas separadas, los distintos vectores tímidos obtenidos.

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 puntos) Un nodo de un árbol binario de enteros se dice que es *singular* si la suma de los valores almacenados en sus nodos antepasados es igual a la suma de los valores almacenados en sus nodos descendientes. Implementa un subprograma que, dado un árbol binario de enteros, devuelva el número de nodos singulares que tiene.

Aparte de implementar este subprograma, debes indicar la complejidad del mismo.

4. (3 puntos) Nos han encargado implementar un sistema para la gestión de la admisión en el Servicio de Urgencias de un Hospital. Cuando un paciente llega al servicio, se le toman los datos y se le asigna un código de identificación único (un número entero no negativo). A partir de ahí, el paciente espera a ser atendido. El orden de atención da prioridad a los pacientes por orden de llegada. Una vez atendido un paciente, sus datos se eliminan del sistema. Así mismo, en cualquier momento un paciente puede desistir de ser atendido. En este caso, en el control de salida se le solicita su número de identificación, y se elimina todo rastro de él del sistema.

La implementación del sistema se deberá realizar como un TAD `GestionAdmisiones` con las siguientes operaciones:

- `crea()`: Operación constructora que crea un sistema de gestión de admisiones vacío.
- `an_paciente(codigo, nombre, edad, sintomas)`: añade al sistema un nuevo paciente con código de identificación `codigo`, con nombre `nombre`, con edad `edad` y con una descripción de síntomas `sintomas`. En caso de que el código esté duplicado, la operación señala un error *"Paciente duplicado"*.
- `info_paciente(codigo, nombre, edad, sintomas)`: `nombre`, `edad` y `sintomas` devuelven la información correspondiente al paciente con código `codigo`. En caso de que el código no exista, levanta un error *"Paciente inexistente"*.
- `siguiente(codigo)`: almacena en `codigo` el código del siguiente paciente a ser atendido. En caso de que no haya más pacientes, esta operación levanta un error *"No hay pacientes"*.
- `hay_pacientes()`: devuelve `true` si hay más pacientes en espera, y `false` en otro caso.
- `elimina(codigo)`: elimina del sistema todo el rastro del paciente con código `codigo`. Si no existe tal paciente, la operación no tiene efecto.

Dado que este es un sistema crítico, la implementación de las operaciones debe ser lo más eficiente posible. Por tanto, debes elegir una representación adecuada para el TAD, implementar las operaciones y justificar la complejidad resultante.