Estructuras de Datos y Algoritmos
Grado en Ingeniería Informática

Segundo Examen Parcial, Junio de 2012.

1. (2 puntos) Diseñar recursivamente, y sin usar estructuras de datos auxiliares ni el iterador de la clase, un método de la clase `Arbus` que dada una clave k que se sabe está en el árbol, devuelva la siguiente clave en orden creciente. Calcular el coste de dicho método.

2. (4 puntos) Se desea un TAD *Consultorio* que simule el comportamiento de un consultorio médico simplificado. Dicha especificación hará uso de los TADs *Medico* y *Paciente*, que se suponen ya conocidos. Las operaciones del TAD *Consultorio* son las siguientes:

- *ConsultorioVacio*: crea un nuevo consultorio vacío.
- *nuevoMedico*: da de alta un nuevo médico en el consultorio.
- *pideConsulta*: un paciente se pone a la espera de ser atendido por un médico.
- *pideConEnchufe*: igual que la anterior, pero el paciente será atendido por delante del resto de pacientes.
- *tienePacientes*: informa de si un médico tiene pacientes esperando.
- *siguientePaciente*: consulta el paciente al que le toca el turno para ser atendido por un médico dado.
- *atiendeConsulta*: elimina el siguiente paciente de un médico.
- *numCitas*: indica el número de citas que un mismo paciente tiene en todo el consultorio.

Se pide:

1. La cabecera de cada operación, indicando además si es generadora, observadora o modificadora, y si es total o parcial.
2. La definición de la representación elegida para el TAD.
3. El coste esperado para cada operación con esa representación.
4. La implementación con todo detalle de las operaciones *pideConsulta*, *atiendeConsulta* y *numCitas*.

3. (4 puntos) Implementar una función que encuentre la forma *más rápida* de viajar desde una casilla de salida hasta una casilla de llegada de una rejilla. Cada casilla de la rejilla está etiquetada con una letra, de forma que en el camino desde la salida hacia la llegada se debe ir formando (de forma cíclica) una palabra dada. Desde una celda se puede ir a cualquiera de las cuatro celdas adyacentes.

Como ejemplo, a continuación aparece la forma más corta de salir de una rejilla de 5×8 en la que el punto de salida está situado en la posición (0, 4) y hay que llegar a la posición (7, 0) y la palabra que hay que ir formando por el camino es `EDA`.

0	M	D	A	A	E	E	D	A
1	A	E	E	D	D	A	N	D
2	D	B	D	X	E	D	A	E
3	E	A	E	D	A	R	T	D
4	E	D	M	P	L	E	D	A
	0	1	2	3	4	5	6	7

Para la implementación puedes suponer la existencia de dos variables globales `N` y `M` que determinan el tamaño de la rejilla, así como la información de la propia rejilla,

```
char lab[N][M];
```

También puedes suponer la existencia de una variable global que contiene la palabra a utilizar,

```
Lista<char> palabra;
```

Las función recibirá el punto origen y el punto destino y deberá determinar la forma más rápida de viajar de uno a otro (si es que esto es posible), dando las direcciones que hay que ir cogiendo (en el caso del ejemplo será E, N, E, E, E, etc.). Si necesitas parámetros adicionales, añádelos indicando sus valores iniciales.