Código: 75.001

Asignatura: Estructura de la Información

Curso: 2010/2011 – segundo semestre

PEC 1

Ejercicio 1 [2 puntos]

Una cooperativa de taxis decide hacer una aplicación para gestionar la asignación de los servicios a sus taxistas. Se desea hacer las asignaciones en el mismo orden que se han generado. Es decir, si en el momento que se solicitan servicios no hay ningún taxi disponible, en cuanto un taxista quede libre tomará el primer servicio pedido. El mismo criterio se desea aplicar a los taxistas. En caso de que haya taxistas libres, en el momento que se pida un servicio, se asignará al taxista que haga más tiempo que haya entregado.

El TAD *TaxiManagement* deberá tener la siguiente funcionalidad:

- Obtener el taxi libre con mayor tiempo de espera.
- Dar de alta un servicio pedido por un cliente.
- Asignar el servicio más antiguo en el taxi libre con más tiempo de espera.
- Dar por concluido un servicio.

Apartado 1.1) Dad la signatura del TAD *TaxiManagement*. Es decir, indicad el nombre que daríais a las operaciones encargadas de cada funcionalidad requerida, indicando también, cuales deberían ser los parámetros de entrada y cuál la salida en caso de que se necesitara. Suponed que existen los TADs *Taxi* para gestionar los datos de un taxi y *Service* para gestionar los datos de un servicio de taxi.

Apartado 1.2) El TAD específico que se pide estará compuesto por TADs genéricos de la biblioteca de la asignatura. Indica cuáles y cuántos serán necesarios para poder llevar a cabo la gestión.

Apartado 1.3) Indica qué operaciones de los TADs de biblioteca de la asignatura que has elegido deberás llamar para resolver las operaciones del TAD *TaxiManagement*. Es decir, suponiendo que se le pidiera la funcionalidad de añadir el número de teléfono desde donde se ha llamado y el TAD escogido fuera una lista, una posible respuesta sería: ***añadirAlfinal(teléfono)***. Si la funcionalidad fuera eliminar el número de teléfono, una posible respuesta sería: **obtener un recorrido con la operación *posiciones*(). Iterar por las diferentes posiciones hasta encontrar la posición donde se encuentra el teléfono buscado. Con la posición encontrada llamar a *borrar(posicion)***.

Apartado 1.4) Haz la especificación contractual del TAD. Utiliza de modelo la especificación del apartado 1.2.3 del Módulo 1 de los materiales docentes. Se valorará especialmente la concisión (ausencia de elementos redundantes o innecesarios), precisión (definición correcta del resultado de las operaciones), completitud (consideración de todos los

casos posibles en que se puede ejecutar cada operación) y falta de ambigüedades (conocimiento exacto de cómo se comporta cada operación en todos los casos posibles) de la solución. Es importante responder a este apartado usando una descripción condicional y no procedimental. La experiencia nos demuestra que no siempre resulta fácil distinguir entre ambas descripciones, es por ello que hacemos especial hincapié insistiendo que pongáis mucha atención en vuestras definiciones. A título de ejemplo indicaremos que la descripción condicional (la correcta a usar en el contrato) de *llenar un vaso vacío con agua* seria:

> @pre el vaso se encuentra vacío.
> @post el vaso se encuentra lleno de agua

En cambio una descripción procedimental (incorrecta) tendría una forma similar a:

> @pre el vaso debería encontrarse vacío, si no se encontrara vacío debería vaciarse.
> @post Se acerca el vaso al grifo y se echa agua hasta que esté lleno

Debéis también tener en cuenta que un contrato debería disponer de invariante siempre que esta fuera necesaria para describir el TAD.

Ejercicio 2 [2 puntos]

Apartado 2.1) Escribe un algoritmo que permita contabilizar cuántas veces se repite un vector de caracteres en el contenido de un otro vector. Ejemplo: sea el vector v1 = {u, n, , v, e, c, t, o, r, , d, e, , c, a, r, a, c, t, e, r, e, s, , d, e, , c, u, a, l, q, u, i, e, r, , t, i, p, o} y el vector v2 = {d, e}. Si se pide cuantas veces se repite v2 en v1, la respuesta deberá ser 2 veces.

Apartado 2.2) Haz un estudio del coste temporal del algoritmo.

Ejercicio 3 [2 puntos]

Indica qué árbol tiene los siguientes recorridos en preorden {2,5,1,7,3,9,4,12} y en inorden {5,7,1,3,2,9,12,4}. Razona la respuesta explicando cómo llegas a la conclusión.

Ejercicio 4 [2 puntos]

Apartado 4.1) Explicad con vuestras palabras las diferencias entre los TADs *Cola* y *Lista*. ¿En qué casos usaríais una cola y en qué otros una lista?

Apartado 4.2) Indica si la afirmación: "El coste asintótico describe a la perfección el rendimiento de una operación" es cierta o falsa, razonando tu respuesta.

Apartado 4.3) ¿Se puede buscar un elemento de una lista ordenada con repeticiones con un coste logarítmico? ¿Y si la lista no permite repeticiones?

Apartado 4.4) Explica brevemente cómo se relacionan los conceptos de "recursividad" y "recorridos de un árbol".

Ejercicio 5 [2 puntos]

Apartado 5.1) Se pide que implementéis el TAD *PilaUnica*, que extiende del TAD *Pila* de la biblioteca pero que asegura cumplir siempre la invariante:

@invariante Todos sus elementos son diferentes.

Formalmente: $all (e1: pila,! $ exists (e2: pila, e1 = e2));

NOTA: la implementación debéis hacerla usando herencia. Es decir, solamente hace falta implementar las operaciones nuevas o las ya existentes que habría que cambiar.

Apartado 5.2) ¿Qué coste tendría la operación de empilar un elemento?

Apartado 5.3) ¿Qué estructura usarías para mejorar su rendimiento?