UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

E.T.S.E

PROGRAMACIÓN II

Examen

Profesor: David E. Losada **Mayo 2016**

Nombre y apellidos:

1. (1.5 puntos) En un TAD EnterosLargos disponemos de una serie de funciones para realizar una serie de operaciones aritméticas con enteros largos. La cabecera de estas funciones, tal y como está expresada en el **fichero .h**, es la siguiente:

```
void crea(EnteroLargo *e, unsigned int numero_digitos);
void suma (EnteroLargo e1, EnteroLargo e2, EnteroLargo *resultado);
void resta (EnteroLargo e1, EnteroLargo e2, EnteroLargo *resultado);
void multiplica (EnteroLargo e1, EnteroLargo e2, EnteroLargo *resultado);
void divisionentera (EnteroLargo e1, EnteroLargo e2, EnteroLargo *resultado);
void restodivisionentera (EnteroLargo e1, EnteroLargo e2, EnteroLargo *resultado);
void libera(EnteroLargo *e)
int mayorque(EnteroLargo e1, EnteroLargo e2)
```

a) completa la definición de la línea typedef del tipo EnteroLargo **tal y como debería figurar en el .h**

typedef ..... EnteroLargo

Explica por qué debe ser así esa definición y qué implicaciones tiene para el usuario del TAD y para el programador del TAD.

b) desde un programa que quiere usar la librería, se han definido tres variables locales:

EnteroLargo e1, e2, e3, e4;

escribe las instrucciones habría que escribir para, en esas tres variables, crear tres enteros largos (e1 y e2 con 100 dígitos; e3 y e4 con 200 dígitos)

c) escribe las instrucciones que habría que escribir para que en la variable e3 quedase almacenado el valor de 2*e1 + e2 (sugerencia: utilizad la variable e4 para almacenar algún resultado intermedio)

d) si, tras las instrucciones anteriores, queremos reutilizar la variable e1 para crear en ella otro enterolargo con capacidad para 1000 dígitos, qué instrucciones tendríamos que escribir?

e) si el programador del TAD cambia la implementación del mismo para incluir una mejora en la rapidez de alguna de las operaciones, explica en qué debería cambiar el programa anterior para acomodarse a esos cambios.

2. (1.2 puntos) En términos de **orden superior de complejidad <u>temporal</u>**, tenemos un algoritmo sublineal (A), un algoritmo cúbico (B), un algoritmo logarítmico (C) y un algoritmo exponencial (D). Todos ellos resuelven el mismo problema.

   a. Ordénalos de mejor a peor

   b. Indica de los cuatro algoritmos cuales son "usables" y cuales, por sus características de complejidad, difícilmente sirven para algo. Explica (1 o 2 frases como máximo) por qué.

   c. Respecto al mejor de los cuatro algoritmos: ¿es posible que en alguna situación sea más lento que alguno de los otros? ¿cuándo es previsible que pueda pasar esto?

   d. Razona sobre la siguiente cuestión: en términos de **orden superior de complejidad <u>espacial</u>** si el algoritmo C es cuadrático y el algoritmo A es exponencial. ¿cuál preferiremos en esta situación? ¿el que es mejor en términos de complejidad temporal o el que es mejor en términos de complejidad espacial? ¿por qué?

3. (2 puntos) Se dispone de un TAD lista con las operaciones típicas (ver .h a continuación) y donde el tipoelemento es un struct que permite guardar nombres de personas junto con sus edades.

```
struct Datos {
    char nombre[100];
    unsigned short edad;
};

typedef struct Datos TIPOELEM;

typedef void * POSICION;

typedef void * TLISTA;
void crea(TLISTA *l);
void destruye(TLISTA *l);
POSICION primero(TLISTA l);
POSICION fin(TLISTA l);
POSICION siguiente(TLISTA l, POSICION p);
POSICION anterior(TLISTA l, POSICION p);
int esVacia(TLISTA l);
void recupera(TLISTA l, POSICION p, TIPOELEM *e);
int longitud(TLISTA l);
void inserta(TLISTA *l, POSICION p, TIPOELEM e);
void suprime(TLISTA *l, POSICION p);
void modifica(TLISTA *l, POSICION p, TIPOELEM e);
```

Escribe una función que, utilizando el TAD anterior, reciba una lista de personas (parámetro de entrada) e informe (mediante 2 parámetros de salida) de cuántas personas hay menores de edad y cuántas personas hay en edad de jubilación (mayores que 65 años).

4. (0.5 puntos) ¿qué complejidad algorítmica temporal tiene la función anterior (la del ejercicio 3) en el mejor de los casos, en el peor de los casos y el caso promedio? ¿por qué?

5. (0.5 puntos) Una función que consistiese en determinar si hay algún jubilado en la lista, ¿qué complejidad temporal tendría en el mejor de los casos, en el peor de los casos y el caso promedio? ¿por qué?

6. (0.6 puntos) ¿y una función que consistiese en determinar si hay 3 menores de edad en la lista?

7. (0.9 puntos) Dado el vector de 6 elementos que figura a continuación, completa la tabla reflejando las tres primeras modificaciones (permutaciones de la lista original) que sufre el vector si lo ordenamos con los métodos de la burbuja, de selección y de inserción. La ordenación a realizar es de menor a mayor.

Método de burbuja:

14	6	3	23	2	7

Método de selección:

14	6	3	23	2	7

Método de inserción:

14	6	3	23	2	7

8. (1.2 puntos) Si buscamos, mediante búsqueda binaria, el elemento 42 en el vector

1	3	6	8	9	52	67

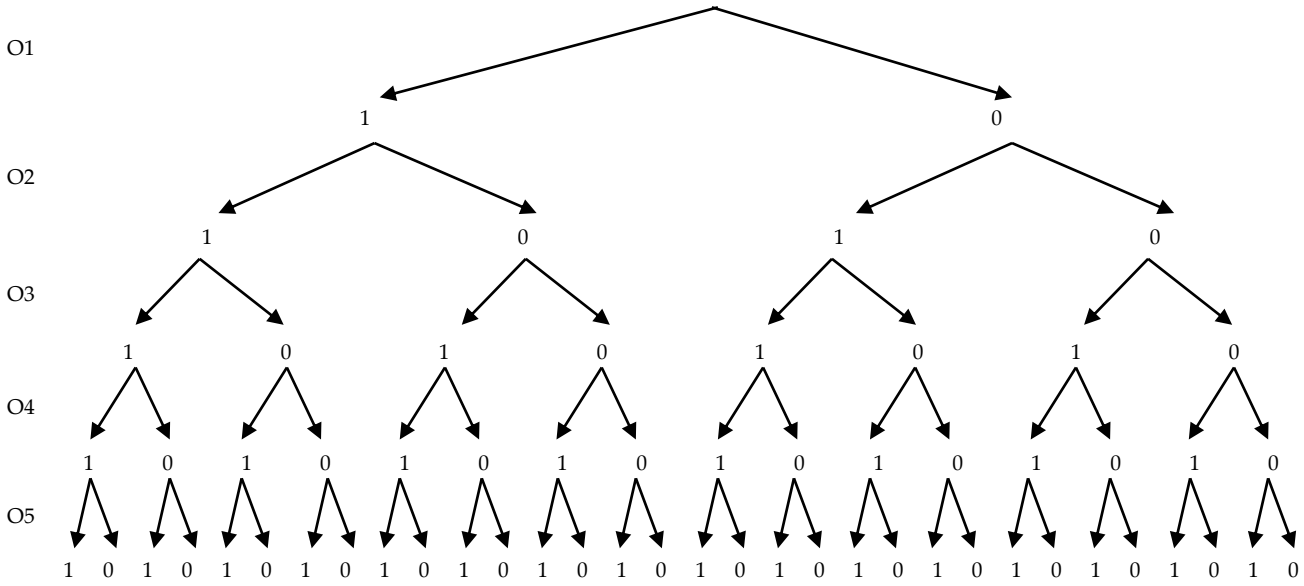
a. ¿cuál es el primer elemento con el que compara el buscado? (indica tanto la posición que ocupa en el vector –las posiciones comienzan a contarse en la 0- como el valor que tiene)

b. ¿cuántas comparaciones con elementos del vector realiza el método hasta que llega a determinar que el elemento 42 no figura en el vector?

c. la búsqueda binaria es un caso de estrategia divide y vencerás, ¿cuál es el valor de k (número de subproblemas a resolver)? ¿cuál es el valor de c –constante por la que se divide el tamaño del problema en cada recursión- ?

9. (1.6 puntos) El siguiente árbol representa el espacio de búsqueda de un problema de mochila entera donde se contemplan 5 posibles objetos (O1, O2, O3, O4 y O5) para una mochila que puede cargar un **máximo de 30 kgs**. La siguiente tabla detalla el peso y valor de cada objeto:

Objeto	Peso	Valor
O1	6	30
O2	10	40
O3	20	60
O4	5	10
O5	15	20



a) si utilizásemos una estrategia de fuerza bruta:
   1. ¿cuántos nodos en el árbol anterior serían explorados? (sin contar el nodo raíz)
   2. ¿qué valor y qué objetos nos llevaríamos en la mochila?
   3. ¿sería viable la estrategia si tenemos más objetos que cinco? ¿por qué?

b) si utilizásemos una estrategia voraz guiada por una heurística de selección de objetos que toma siempre el objeto de mayor valor por unidad de peso:
   1. ¿qué objetos nos llevaríamos y qué valor tendríamos en total en la mochila?
   2. ¿sería viable esta solución computacional voraz si crece mucho el número de objetos? ¿por qué?

c) en el árbol anterior, si lanzamos una estrategia de exploración en profundidad con vuelta atrás cuando encontramos un camino no factible (esto es, excedemos peso máximo de la mochila)
   1. marca en el dibujo de árbol (con una cruz) aquellos puntos donde se corta por ser el camino no factible. Para realizar esto, supón que el proceso para cuando se encuentra la primera solución (poned sólo los cortes que hace el algoritmo antes de encontrar la primera solución).
   2. ¿qué solución nos da el algoritmo en primera instancia? (qué objetos se introducen y qué valor nos llevaríamos en la mochila)
   3. si tras esa primera solución forzamos una vuelta atrás para mejor solución, ¿qué segunda mejor solución nos daría el algoritmo? (qué objetos se introducen y qué valor nos llevaríamos en la mochila)