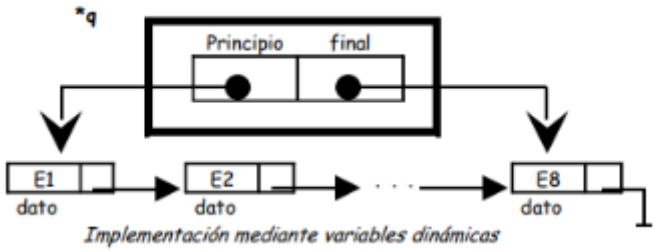
Examen PROGRAMACIÓN II Mayo 2022

1. (1.75 puntos) Escribe la especificación formal de un TAD Bono_Bus que permita crear un bono, insertar dinero en el bono (número total de euros), saber si ha dinero en el bonobús y realizar un gasto de un viaje de bus (siempre cuesta 1 euro).

   Parte sintáctica correcta vale 0.75 puntos. Errores en la etiquetación de funciones constructoras restan hasta 0.25 puntos.
   Parte semántica correcta vale 1 punto. Cada operación especificada semánticamente de modo correcto suma 0.5 puntos.

2. (1 punto) Una cola se implementa con la estructura del siguiente gráfico:



Implementación mediante variables dinámicas

   Las declaraciones dentro de cola.c son las siguientes:

```
typedef int TELEMENTO;
typedef struct nodo{
        TELEMENTO dato;
        struct nodo *sig;
} TNodo;
typedef struct {
        TNodo *principio
        TNodo *final;
} STCOLA;
typedef STCOLA *TCOLA;
```

   Cada apartado cuenta 0.5 puntos, errores de sintaxis restan hasta 0.25 puntos y errores lógicos graves hasta 0.5 puntos.

   a. Escribe la implementación de la función que elimina el primer elemento de la cola
   b. Escribe la implementación de la función que inserta un elemento al final de la cola

3. (3 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, DNIs o NIEs, nacionalidad (que es una cadena de caracteres, por ejemplo, “estadounidense”) y un campo vacunado, que indica el numero de vacunas que ha recibido cada persona.

```
struct Datos {
   char nombre[100];
   char dni_nie[9];
   char nacionalidad[100];
   unsigned short edad;
   unsigned short vacunado;
}

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);
```

Por otro lado, se dispone de un **TAD cola con prioridad** (ver .h a continuación) que trabajaría también con el mismo tipo de elementos de la lista.

```
typedef void *TCOLAPRIO;
void crearColaPrio(TCOLAPRIO *q);
void destruirColaPrio(TCOLAPRIO *q);
int esColaVaciaPrio(TCOLAPRIO q);
void consultarPrimerElementoColaPrio(TCOLAPRIO q, TIPOELEM *e);
void suprimirElementoColaPrio(TCOLAPRIO *q);
void anadirElementoColaPrio(TCOLA *q, TIPOELEM e, unsigned int prioridad);
```

donde el número de prioridad es un valor mayor o igual a 1, personalizable por el software que use esta librería, y que cuando mayor valor tenga de prioridad el elemento que insertemos más rápidamente saldrá el elemento de la cola.

El SERGAS dispone de una lista con todas las personas domiciliadas en territorio gallego donde se almacena cuantas vacunas ha recibido cada persona. En estos momentos la vacunación de la tercera dosis está mayoritariamente completa en el conjunto de la población. Se desea convocar a todas las personas mayores (edad superior a 80 años) para una cuarta inyección. Además, todas aquellas personas que tengan menos de 3 vacunas (independientemente de su edad) deben también ser llamadas, aunque sean menos prioritarias para vacunarse que los mayores. Los mayores de 80 con menos de 3 vacunas con los más prioritarios de todo pues están más expuestos al virus que los de 80 que ya tienen 3 vacunas. Las personas en cada grupo de prioridad deben incorporarse a la vacunación en el orden en el que están en la lista del SERGAS. La lista del SERGAS pudiera estar vacía, por lo que la función implementada deberá considerar todos los cosos posibles.

Escribe una función que, utilizando las librerías anteriores, reciba esa lista de personas (parámetro de entrada) y genere una cola con prioridad que sería la que fuese utilizada en el proceso de vacunación en un vacunódromo.

Criterios de corrección: La incorrecta definición de la cabecera de la función (parámetros de entrada y salida) resta 0.5 puntos. Cada expresión errónea (por ejemplo, invocar a una función poniendo incorrectamente los argumentos (p.e. poner un LISTA* cuando debe ser un LISTA, incorporar incorrectamente &s o *s donde no corresponde)) resta 0.5 puntos. Errores lógicos graves (p.e. lazos infinitos, potenciales fallos de memoria, segmentation fault, etc) restan hasta 2 puntos. La incorrecta implementación de la funcionalidad requerida resta hasta 3 puntos. La incorrecta definición del número de prioridades requerido resta 1 punto.

4. (1 punto) Dada la lista del SERGAS descrita en el apartado anterior, si tuviésemos la necesidad de conocer cuántos mayores tienen menos de 3 vacunas, ¿qué complejidad algorítmica temporal tendrá el correspondiente proceso de cómputo de ese número? Explica cuál sería la complejidad temporal en el mejor de los casos, en el peor de los casos y en el caso promedio.

5. (1 punto) Una nación extranjera desea conocer si algún ciudadano de su país está en la lista del SERGAS y tiene menos de 3 vacunas. En la implementación más eficiente posible de este proceso, explica cuál sería la complejidad temporal en el mejor de los casos, en el peor de los casos y el caso promedio.

6. (1.25 puntos) Analiza brevemente la búsqueda secuencial y la búsqueda binaria. Explica i) qué tipo de técnicas algorítmicas son, ii) que complejidad inferior y superior tienen, iii) qué prerrequisitos tienen para ser utilizadas, iv) cuál utilizarías para poner en un sistema en producción y v) indica también si la mejor solución que propones sería siempre la más rápida. (0.25 por una explicación correcta de cada uno de estos aspectos).

7. (1 punto) El gráfico de debajo muestra la complejidad computacional de una solución divide y vencerás en función de los parámetros k, c e i.
   - Si $k < c^t$ entonces $t(n) \in O(n^t)$.
   - Si $k = c^t$ entonces $t(n) \in O(n^t \log n)$.
   - Si $k > c^t$ entonces $t(n) \in O\left(n^{\log_c k}\right)$.

   a. Explica el significado de k, c e i (indicando también si queremos que cada uno de ellos tenga un valor alto o bajo y por qué).
   b. En las dos variantes de multiplicación de enteros grandes (x multiplicado por y) descritas debajo, indica cual es más eficiente, justificando porqué en referencia a sus parámetros divide y vencerás.

Variante 1

$$xy = ac\, i^n + (ad + bc) i^{n/2} + bd$$

Variante 2

$$xy = ac\, i^n + \big((a - b)(d - c) + ac + bd\big) i^{n/2} + bd$$