EXAME PROGRAMACIÓN II MAIO 2022 CONVOCATORIA ORDINARIA

1. **(1.75 puntos)** Escribe a especificación formal dun `TAD-bonobus` que permita crear un bono, insertar diñeiro no bono (número natural de euros), coñecer se hai diñeiro no bono e realizar un gasto de viaxe no bus (cuxo prezo é sempre dun euro).

Parte sintáctica correcta: `0.75 puntos`. Erros na etiquetación de función restan ata `0.25 puntos`
Parte semántica correcta vale `1 punto`. Cada operación especificada de modo correcto suma `0.5 puntos`.

2. **(1 punto)** Unha cola impleméntase internamente coa estructura descrita no seguinte gráfico:

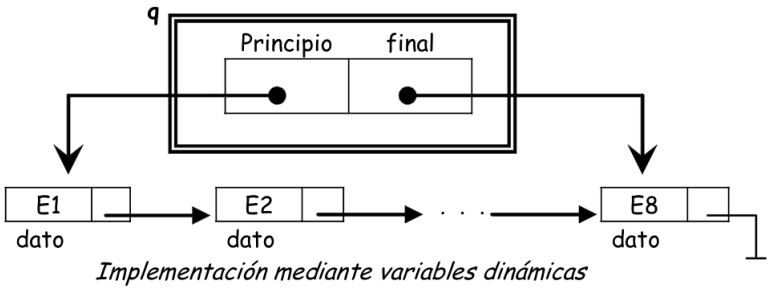

Figura 1: Implementación dunha cola mediante variables de tipo punteiro

As declaracións dentro de `cola.c` son as seguintes:

```
typedef int TELEMENTO;
typedef struct nodo {
     TELEMENTO datos
     struct nodo *sig;
} TNodo;

typedef struct {
    TNodo *principio, *final;
} STcola;

typedef STcola * TCOLA;
```

Cada apartado conta `0.5 puntos`, erros de sintaxe restan ata `0.25 puntos` e erros lóxicos graves ata `0.5 puntos`.

(a) **(0.5 puntos)** Escribe a implementación (con punteiros) da función que elimina o primeiro elemento da cola.

(b) **(0.5 puntos)** Escribe a implementación da función que inserta un elemento ó final da cola.

3. **(3 puntos)** Disponse dun `TAD lista` coas aoperacións típicas (ver `.h` a continuación) e onde o `tipoelemento` é un struct que permite gardar nomes de persoas xunto ás súas idades, DNI, nacionalidade (que é unha cadea de caracteres) e un campo vacinado que indica o número de vacinas que recibeu a persoa.

```
struct Datos{
   char nome[100];
   char dni[9];
```

```
    char nacionalidade[100];
    unsigned short edad;
    unsigned short vacunado;
}
typedef struct Datos TIPOELEM;
typedef void *POSICION;
typedef void *TLISTA;
void crea(TLISTA *l);
void destrue(TLISTA *l);
POSICION primero(TLISTA *l);
POSICION fin(TLISTA *l);
POSICION siguiente(TLISTA *l, POSICION p);
POSICION siguiente(TLISTA *l, POSICION p);
int esVacia(TLISTA *l);
void recupera(TLISTA *l, POSICION p, TIPOELEM e);
void inserta(TLISTA *l, POSICION p, TIPOELEM e);
void modifica(TLISTA *l, POSICION p, TIPOELEM e);
void suprime(TLISTA *l, POSICION p);
int longitud(TLISTA l);
```

Por outro lado, disponse dun `TAD cola` con prioridade que traballaría sobre os elementos da lista creada anteriormente.

```
typedef void *TCOLAPRIO;
void CrearColaPrio(TCOLAPRIO *q);
void DestruirColaPrio(TCOLAPRIO *q);
int esColavaciaPrio(TCOLAPRIO q);
void ConsultarPrimerElemento(TCOLAPRIO *q, TIPOELEMENTO *e);
void suprimirElemento(TCOLAPRIO *q);
void anhadirElementoCola(TCOLAPRIO *q, TIPOELEMENTO e, unsigned int prio);
```

donde o numero de prioridade é un valor maior ou igual a un, personalizable polo software que empregue a librería, e que canto maior valor teña a prioridade do elemento insertado máis rapidamente sairá o elemento da cola.

O **SERGAS** dispón de unha lista con todas as posibles persoas domiciliadas no territorio galego donde se almacena cántas vacinas recibiu cada persoa. Nestos momentos, coa vacinación da terceira xa practicamente completada, desexase convocar a todas as persoas maiores (de idade superior a 80 anos) para unha cuarta inxeccion. Ademáis, todas aquelas persoas que dispoñan de menos de tres vacinas (independentemente da súa idade) deben tamén ser chamadas, ainda que serán chamadas menos prioritarias que as de maiores de 80 anos.

Os maiores de 80 anos con menos de tres vacinas son os máis prioritarios. As persoas en cada grupo de prioridade deben incorporarse á vacinación na orde na que están nas listas do sergas. Cabe contemplar que a lista do serga podería estar vacía, polo que a función deberá contemplar todos os casos posibles.

Escribe unha función que, utilizando as librerías anteriores, reciba esa lista de persoas (parámetro de entrada) e xenere unha lista de prioridade que sería a que fora ser empregada no proceso de vacinación do vacunódromo.

4. **(1 punto)** Dada a lista do SERGAS descrita no apartado anterior si tivésemos a necesidade de coñecer cántos maiores teñen menos de tres ¿que complexidade algorítmica temporal tería o correspondente

proceso de cómputo dese número? Explica cál seria a complexidade temporal no mellor dos casos, no peor dos casos e no caso promedio.

5. **(1 punto)** Unha nación estranxeira desexa coñecer se algún cidadán do seu país está as listas do SERGAS e ten menos de tres vacinas. Na implementación máis eficiente posible deste proceso, explica cal sería a complexidade temporal no mellor dos casos, no peor dos casos e no caso promedio.

6. **(1.25 puntos)** Analiza brevemente a búsqueda secuencial e a búsqueda binaria. Explica os seguintes conceptos:
   (a) **(0.25 puntos)** O tipo de técnicas algorítmicas que son.
   (b) **(0.25 puntos)** A complexidade superior e inferior que teñen.
   (c) **(0.25 puntos)** Os prerrequisitos antes de ser empregadas.
   (d) **(0.25 puntos)** Cál empregarías para poñer en marcha un sistema en produccion.
   (e) **(0.25 puntos)** Indica se a solución proposta sería sempre a máis rápida

7. **(1 punto)** O gráfico de abaixo mostra a complexidade computacional dunha solución divide e vencerás en función de parámetos k, c e i.

   1. Se $k < c^t$, entón $t(n) \in \mathbb{O}(n)^t$
   2. Se $k = c^t$, entón $t(n) \in \mathbb{O}(n^t log(n))$
   3. Se $k < c^t$, entón $t(n) \in \mathbb{O}(n^{log_c k})$

   (a) **(0.5 puntos)** Explica o significado de k, c e i (indicando se deles queremos valores altos ou baixos).
   (b) **(0.5 puntos)** Nas dúas variantes de multiplicación de enteiros grandes (x multplicado por y) descritas debaixo, indica cál é a máis eficiente, xustificando por qué en referencia a parámetros divide e vencerás.

Variante 1

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

Variante 2

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