**Introducción a la Computación**
Primer cuatrimestre de 2016
**Segundo Parcial**

*Departamento de Computación*
*Facultad de Ciencias Exactas y Naturales*
*Universidad de Buenos Aires*

| | Nombre y Apellido: | Nota: | | |
|---|---|---|---|---|
| ▷ Resolver cada ejercicio en hojas separadas.<br>▷ Poner nombre en todas las hojas.<br>▷ Poner LU y nombre en el enunciado.<br>▷ El parcial se aprueba con 65 puntos. | Libreta Universitaria: | Ej. 1 | Ej. 2 | Ej. 3 |

**Ejercicio 1** (30 pts)**.** Dada una lista $L$, se dice que $L$ tiene un elemento mayoritario si existe un valor que se repite más de $\lfloor \frac{|L|}{2} \rfloor$ veces. Se pide escribir en pseudocódigo un algoritmo de *Divide & Conquer* que dada una lista no vacía de enteros positivos devuelva su elemento mayoritario si es que existe, y si no devuelva $-1$. La complejidad del algoritmo propuesto(en el peor caso) debe ser de $\mathcal{O}(|L| \times \log |L|)$. A continuación se exhiben posibles entradas y el resultado esperado:

- `[6]` $\mapsto$ `6`
- `[1,4]` $\mapsto$ `-1`
- `[12,2,7,7]` $\mapsto$ `-1`
- `[13,2,7,13,13]` $\mapsto$ `13`
- `[5,5,5,2,2,5]` $\mapsto$ `5`

Justificar por qué el algoritmo propuesto tiene el orden pedido.

**Ejercicio 2** (35 pts)**.** Sea el TAD `MatrizBinaria` definido de la siguiente manera:

TAD `MatrizBinaria`

- `M.cantFilas()` $\rightarrow \mathbb{Z}$: devuelve la cantidad de filas de `M`
- `M.cantColumnas()` $\rightarrow \mathbb{Z}$: devuelve la cantidad de columnas de `M`
- `M[i,j]` $\rightarrow \{0,1\}$: devuelve el elemento de la posición $(i,j)$ de `M`
- `M.marcar(i,j)`: pone una marca en `M[i,j]`
- `M.desmarcar(i,j)`: deja sin marcas `M[i,j]`
- `M.estáMarcada(i,j)` $\rightarrow \mathbb{B}$: devuelve `True` si `M[i,j]` está marcada, y `False` en caso contrario

donde `M`: `MatrizBinaria` y `i,j`: $\mathbb{Z}$.
Las operaciones `M[i,j]`, `M.marcar(i,j)`, `M.desmarcar(i,j)` y `M.estáMarcada(i,j)` tienen como precondición $0 \leq i <$ `M.cantFilas()` y $0 \leq j <$ `M.cantColumnas()`.

Dada una `MatrizBinaria M`, definimos un *camino de alternancias* como una secuencia de posiciones de `M` que cumple estas condiciones:

- Comienza en la posición superior izquierda de la matriz (`M[0,0]`) con valor 0.
- Pasa como máximo una vez por cada posición de la matriz.

- Alterna siempre $0s$ y $1s$: 010101010101....
- Pasa sólo de una posición en `M` a otra posición adyacente, ya sea vertical u horizontalmente.

Se pide dar en pseudocódigo un algoritmo de *Backtracking* que, dados una `MatrizBinaria M` y dos enteros `i,j`, determine si existe un camino de alternancias que termine en `M[i,j]`. Puede suponerse que $0 \leq$ `i` $<$ `M.cantFilas()` y $0 \leq$ `j` $<$ `M.cantColumnas()`.

Ejemplo: A continuación se muestran dos matrices binarias de $3 \times 4$. $M_1$ tiene un camino de alternancias que termina en `M[2,3]`, y $M_2$ no:

$$M_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad\qquad M_2 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

**Ejercicio 3** (35 pts)**.** Se cuenta con una implementación del TAD `Lista(ℝ)` con las operaciones y los órdenes indicados a continuación:

<u>TAD `Lista(ℝ)`</u>

- `CrearLista()` $\rightarrow$ Lista($\mathbb{R}$): devuelve una lista vacía en $\mathcal{O}(1)$
- `L.agregar(x)`: agrega `x` al final de `L` en $\mathcal{O}(1)$
- `L[i]` $\rightarrow \mathbb{R}$: devuelve el $i$-ésimo elemento de `L` en $\mathcal{O}(1)$
- `L.tamaño()` $\rightarrow \mathbb{Z}$: devuelve la longitud de `L` en $\mathcal{O}(1)$

donde `x`: $\mathbb{R}$ y `i`: $\mathbb{Z}$. La operación `L[i]` tiene como precondición $0 \leq$ `i` $<$ `L.tamaño()`.

Se desea implementar un TAD `CuentaBancaria` que ofrezca las siguientes operaciones, con los órdenes de complejidad indicados:

<u>TAD `CuentaBancaria`</u>

- `AbrirCuenta()` $\rightarrow$ `CuentaBancaria`: Crea una cuenta bancaria con saldo 0 en $\mathcal{O}(1)$
- `C.Acreditar(x)`: Acredita el monto `x` a la cuenta `C` en $\mathcal{O}(1)$
- `C.Debitar(x)`: Debita el monto `x` de la cuenta `C` en $\mathcal{O}(1)$
- `C.Saldo()` $\rightarrow \mathbb{R}$: Devuelve el saldo de la cuenta `C` en $\mathcal{O}(1)$
- `C.CantidadOperaciones()` $\rightarrow \mathbb{Z}$: Devuelve la cantidad histórica de movimientos (créditos y débitos) de la cuenta `C` en $\mathcal{O}(1)$
- `C.ListarUltimasOperaciones(k)` $\rightarrow \mathbb{R}[\,]$: Lista las últimas `k` operaciones realizadas sobre la cuenta `C` en $\mathcal{O}($`k`$)$

donde `x`: $\mathbb{R}$ y `k`: $\mathbb{Z}$. La operación `C.Debitar(x)` tiene como precondición `C.Saldo()` $\geq$ `x`, y `C.ListarUltimasOperaciones(k)` tiene como precondición $0 \leq$ `k` $<$ `C.CantidadOperaciones()`.

Se pide escribir la estructura de representación del TAD `CuentaBancaria`, su invariante de representación y los algoritmos de las 6 operaciones descriptas arriba. Justificar por qué los algoritmos propuestos cumplen con los órdenes de complejidad indicados.