Algoritmos y Estructuras de Datos I
Turno Mañana

Departamento de Computación – FCEyN – UBA
Primer cuatrimestre de 2018

Nombre y apellido: _____

Nº orden: _____ L.U.: _____ Cant. hojas: _____

??

Segundo parcial – 22/6 - Primer cuatrimestre de 2018

ACLARACIONES: Se permite tener UNA hoja A4 con anotaciones durante el parcial. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos.

Entregar cada ejercicio en una hoja separada, numerada y que incluya el nro. de orden.

Justifique *adecuadamente* cada una de sus respuestas.

Ejercicio 1.[25 puntos]
El siguiente programa calcula, para una secuencia de enteros dada, el máximo valor absoluto de la secuencia.

a) **[6 puntos]**Dar su Diagrama de Control de Flujo.

b) **[7 puntos]**Dar un conjunto de casos de test que maximice el cubrimiento de líneas, señalando claramente las líneas cubiertas por cada test.

c) **[7 puntos]**Dar un conjunto de casos de test que maximice el cubrimiento de ejes (branches), señalando claramente los branches cubiertos por cada test.

d) **[5 puntos]**Señale que lineas o branches no se pueden cubrir y si esto es producto de un defecto, señale cuál y como se podría corregir el programa.

```
    int maximoValorAbsoluto(vector<int> numeros){
L1: int maximo = 0;
L2: for(int i=0; i< numeros.size();i++){
L3:   if(maximo < numeros[i]){
L4:     if(numeros[i] <= 0){
L5:       maximo = numeros[i] * (-1);
        }else{
L6:       maximo = numeros[i];
        }
      }
    }
L7: return maximo;
    }
```

Ejercicio 2.[25 puntos]
Dada la siguiente especificación se pide escribir una implementación cuyo único ciclo respete el invariante I. Justificar todas las decisiones.

Especificación

proc esSumaDePares (in s: $seq\langle\mathbb{Z}\rangle$, out res: Bool) {
  Pre $\{s = s_0 \wedge |s| \ \% \ 3 = 0\}$
  Post $\{res = true \leftrightarrow (\forall i : \mathbb{Z})(0 \leq i < |s| \wedge i \ \% \ 3 = 0 \rightarrow_L s_0[i] + s_0[i+1] = s_0[i+2]\}$
}

$$I = 0 \leq i \leq |s| \wedge i \ \% \ 3 = 0 \wedge_L res = true \leftrightarrow (\forall j : \mathbb{Z})(0 \leq j < i \wedge j \ \% \ 3 = 0 \rightarrow_L s_0[j] + s_0[j+1] = s_0[j+2]$$

Ejercicio 3.[25 puntos]

a) **[20 puntos]** Escribir un programa en C++ (o similar) que dada una secuencia no vacía de enteros, devuelva el índice tal que la cantidad de elementos distintos en la subsecuencia desde la posicion 0, hasta el indice incluida, sea igual a la cantidad de elementos distintos de la subsecuencia desde indice+1 hasta el final. Si el índice no existe el programa debe devolver -1. Si existen varios puede devolver cualquiera .

- $\langle 1, 2, 7, 3, 2, 4 \rangle \mapsto \langle 2 \rangle$
- $\langle 1, 1, 7, 3, 2, 4, 6 \rangle \mapsto \langle 3 \rangle$

b) **[5 puntos]** Justifique la complejidad en peor caso del algoritmo

Ejercicio 4.[25 puntos] Se tiene una matriz rectangular de $F \times C$ que representa una pantalla de *la viborita*, uno de los juegos más clásicos de todos los tiempos. En cada casillero hay un booleano, que vale true si hay un segmento de viborita en el casillero, y false si no.

En particular, se pide implementar el siguiente procedimiento:

`proc Largo` (in F:$\mathbb{Z}$, in C:$\mathbb{Z}$, in T:$seq\langle seq\langle \mathsf{Bool} \rangle\rangle$, in fCabeza:$\mathbb{Z}$, in cCabeza:$\mathbb{Z}$)

Dados un tablero y la posición de la cabeza (primer segmento), esta función debe calcular el largo de la viborita representada.

Se garantiza que dos segmentos de viborita **no adyacentes** nunca se tocan horizontal ni verticalmente, y que la viborita no se cruza consigo misma. Por ejemplo, en estos tableros de 4x4:

	0	1	2	3
0	1	1	0	0
1	0	1	0	0
2	0	1	1	0
3	0	0	1	0

Este tablero representa una viborita válida de largo 6. Su cabeza puede ser (0,0) o (3,2).

	0	1	2	3
0	1	1	0	0
1	0	1	0	0
2	0	1	1	0
3	1	1	1	0

Este tablero no representa una viborita válida porque (2,1) y (3,1) no son adyacentes en la viborita, pero son vecinos.

Se pide que la complejidad en el peor caso del algoritmo propuesto sea a lo sumo lineal respecto del largo de la viborita.

a) **[15 puntos]** Implemente la función pedida considerando que la viborita no puede atravesar los bordes del tablero. Recomendamos el uso de funciones auxiliares.

b) **[5 puntos]** Justifique la complejidad en peor caso del algoritmo.

c) **[5 puntos]** Para justificar con palabras: ¿cómo cambiaría el programa presentado si consideramos que la viborita sí puede atravesar los bordes? (recordar que segmentos no adyacentes no se tocan, tampoco atravesando un borde).