# Introducción a la Programación
# Algoritmos y Estructuras de Datos I

Parcial - 1er cuatrimestre 2023

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

| 1.1 | 1.2 | 2.1 | 2.2 | 3 | 4.1 | 4.2 |
|---|---|---|---|---|---|---|
| | | | | | | |

[redacted] NOCHE .....

Cant. [redacted] ar ésta) .....

El parcial se aprueba con 5 puntos. Entregar cada ejercicio en hoja separada. No se permite consultar ningún material durante el examen.

**Ejercicio 1.** 2 puntos

1. [1 punto]
   Completar en las siguientes especificaciones nombres adecuados para el problema $a$, los parámetros $b$ y $c$, y las etiquetas $x$, $y$, $z$, $u$ y $w$.

   `problema a` (in b: $seq\langle Char \times Char\rangle$, in c: $seq\langle Char\rangle$ ) : $seq\langle Char\rangle$ {
   `requiere x:` $\{ (\forall i,j : \mathbb{Z})((0 \leq i < |b| \land 0 \leq j < |b| \land i \neq j) \rightarrow b[i]_0 \neq b[j]_0\}$
   `requiere y:` $\{ (\forall i,j : \mathbb{Z})((0 \leq i < |b| \land 0 \leq j < |b| \land i \neq j) \rightarrow b[i]_1 \neq b[j]_1\}$
   `requiere z:` $\{ (\forall i : \mathbb{Z})(0 \leq i < |c| \rightarrow (\exists j : \mathbb{Z})(0 \leq j < |b| \land b[j]_0 = c[i]))\}$
   `asegura u:` $\{ |resultado| = |c| \}$
   `asegura w:` $\{ (\forall i : \mathbb{Z})(0 \leq i < |c| \rightarrow (\exists j : \mathbb{Z})(0 \leq j < |b| \land b[j]_0 = c[i] \land b[j]_1 = resultado[i])) \}$
   }

2. [1 punto]
   Especificar el siguiente problema (se puede especificar de manera formal o semi-formal):
   Dados los inputs *b:* $seq\langle Char \times Char\rangle$, *m:* $seq\langle seq\langle Char\rangle\rangle$ y *n:* $seq\langle seq\langle Char\rangle\rangle$, retornar verdadero si $n$ es igual al resultado de aplicar el `problema a` (del punto 1.1) a cada elemento de la secuencia $m$.

**Ejercicio 2.** 4 puntos

1. [2 puntos]
   Programar en Haskell una función que satisfaga la especificación del `problema a` del Ejercicio 1. Recordá escribir los tipos de los parámetros.

2. [2 puntos]
   Programar en Python una función que satisfaga la especificación del `problema a` del Ejercicio 1. Recordá escribir los tipos de los parámetros y variables que uses en tu implementación.

**Ejercicio 3.** 2 puntos
Sea la siguiente especificación del problema `aprobado` y una posible implementación en lenguaje imperativo:

`problema aprobado` (in notas: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$ {
`requiere:` $\{|notas| > 0\}$
`requiere:` $\{(\forall i : \mathbb{Z})(0 \leq i < |notas| \rightarrow 0 \leq notas[i] \leq 10)\}$
`asegura:` $\{result = 1 \leftrightarrow$ todos los elementos de *notas* son mayores o iguales a 4 y el promedio es mayor o igual a 7$\}$
`asegura:` $\{result = 2 \leftrightarrow$ todos los elementos de *notas* son mayores o iguales a 4 y el promedio está entre 4 (inclusive) y 7$\}$
`asegura:` $\{result = 3 \leftrightarrow$ alguno de los elementos de *notas* es menor a 4 o el promedio es menor a 4$\}$
}

```
def aprobado(notas: list[int]) -> int:
L1:   suma_notas: int = 0
L2:   i: int = 0
L3:   while i < len(notas):
L4:      if notas[i] < 4:
L5:         return 3
L6:      suma_notas = suma_notas + notas[i]
L7:      i = i + 1
L8:   if suma_notas >= 7 * len(notas):
L9:      return 2
L10:  else:
L11:     if suma_notas > 4 * len(notas):
L12:        return 2
L13:     else:
L14:        return 3
```

1. Dar el diagrama de control de flujo (control-flow graph) del programa `aprobado`.
2. Escribir un test suite que ejecute todas las líneas del programa `aprobado`.
3. Escribir un test suite que tenga un cubrimiento de **al menos** el 50 por ciento de decisiones ("branches") del programa.
4. Explicar cuál/es es/son el/los error/es en la implementación. ¿Los test suites de los puntos anteriores detectan algún defecto en la implementación? De no ser así, modificarlos para que lo hagan.

**Ejercicio 4.** 2 puntos

1. [1 punto] Suponga las siguientes dos especificaciones de los problemas p1 y p2:

```
problema p1(x:Int)=res:Int {
  requiere A;
  asegura C;
  }
```

```
problema p2(x:Int)=res:Int {
  requiere B;
  asegura C;
  }
```

   Si A es más fuerte que B, ¿Es cierto que todo algoritmo que satisface la especificación p1 también satisface la especificación p2? ¿Y al revés?, es decir, ¿Es cierto que todo algoritmo que satisface la especificación p2 también satisface la especificación p1? Justifique.

2. [1 punto] ¿Es posible que haya un test suite con 100 % de cubrimiento de nodos que todos los test pasen pero que igual el programa tenga un bug? Justifique.