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.

Ejercicio 1. [25 puntos] Completar la precondición de la especificación dada y demostrar formalmente la corre del programa `hacerPositivaPosI` con respecto a su especificación usando la precondición más débil.

Especificación

```
proc hacerPositivaPosI (inout s: seq(Z), in i: Z) {
    Pre {s = s0 ∧ ...}
    Post {length(s) = length(s0) ∧ 0 ≤ i < length(s) ∧L s[i] > 0 ∧
          (∀j : Z)((0 ≤ j < length(s) ∧ i ≠ j) →L s[j] = s0[j])}
}
```

Implementación en SmallLang

```
if (s[i]<0)
  s[i] = -1 * s[i];
else
  skip;
endif
```

Ejercicio 2. [35 puntos] Sea el siguiente ciclo con su correspondiente precondición y postcondición:

P_c : $\{length(s)\ mod\ 2 = 0 \wedge i = length(s) - 1 \wedge suma = 0\}$

```
while (i >= length(s) / 2) do
    suma := suma + s[length(s)-1-i];
    i := i - 1;
endwhile
```

$$Q_c : \left\{ i = length(s)/2 - 1 \ \wedge_L \ suma = \sum_{j=0}^{length(s)/2-1} s[j] \right\}$$

a) **[10 puntos]** Especificar un invariante de ciclo que permita demostrar que el ciclo cumple la postcondición.

b) **[5 puntos]** Especificar una función variante y una cota que permitan demostrar que el ciclo termina.

c) **[20 puntos]** Demostrar formalmente la corrección y terminación del ciclo usando el Teorema del invariante.

Ejercicio 3. [20 puntos] Escribir un programa en C++ que, dado un arreglo de enteros a, devuelva el intervalo m grande de elementos consecutivos que no esté contenido en él. Si todos los elementos de a son consecutivos, debe devolver un arreglo con el elemento siguiente al último de a. Si a es vacío, debe devolverse un arreglo con el 0.
Por ejemplo, si a = [5,3,1,8,9], el programa debería devolver [6,7] y si a = [4,5,6], el programa debería devolver [7].

Ejercicio 4. [20 puntos] Escribir un programa en C++ que, dada una matriz de enteros y un natural n, determine si la matriz hay al menos una escalera de largo n. Llamaremos escalera a una secuencia de elementos consecutivos estrictamen crecientes. Buscaremos escaleras únicamente en filas y en columnas. Por ejemplo, la primera matriz tiene dos escaleras longitud 4 en la primera fila, empezando en la primera posición o en la segunda. La segunda matriz tiene una escalera en segunda columna, y la tercer matriz no tiene escaleras de longitud 4.

1	5	6	7	8
4	2	6	3	3
7	8	3	3	0
4	2	6	4	3
7	8	9	3	3

7	8	9	5	3
4	2	6	4	3
7	8	3	5	0
4	2	6	6	3
7	8	9	7	3

7	3	9	5	1
4	8	9	3	4
6	8	9	0	5
5	9	9	8	4
7	8	9	9	3