Análisis y Diseño de Algoritmos
Ejercicios previos al primer examen parcial

- Demuestra que $n + 10 \in O(n)$.
  Eligiendo $c = 2$ y $n_0 = 10$, probaremos que $n+10 \leq 2n$ para toda $n \geq 10$. Verificamos a mano para $n = 10$, donde tenemos $10 + 10 \leq 2(10)$.
  Suponemos que se cumple para $n = k$ que $k + 10 \leq 2k$ y deseamos demostrar para $n = k + 1$ que $(k + 1) + 10 \leq 2(k + 1)$. Reescribimos como $(k + 10) + 1 \leq (2k) + 2$. Como $k + 10 \leq 2k$ por hipótesis de inducción y $1 \leq 2$, entonces $(k+1)+10 \leq 2(k+1)$ que es lo que se deseaba demostrar.

- Demuestra que $5n \log_2 n \in \Omega(n)$.
  Eligiendo $c = 1$ y $n_0 = 2$, probaremos que $5n \log_2 n \geq n$ para toda $n \geq 2$. Verificamos a mano para $n = 2$, donde tenemos $5(2)(1) \geq 2$.
  Suponemos que se cumple para $n = k$ que $5k \log_2 k \geq k$ y deseamos demostrar para $n = k+1$ que $5(k+1)\log_2(k+1) \geq k+1$. Basta observar que $\log_2(k + 1) \geq 1$ para $k \geq 2$ y que $5(k + 1) \geq k + 1$ para concluir la demostración. El chiste de este problema es elegir la $n_0$ correcta, porque el logaritmo se indetermina para $n = 0$ y es 0 para $n = 1$.

- Demuestra que el siguiente algoritmo para calcular la resta de dos enteros no negativos es correcto:

  **función** $\textsc{Resta}(a, b \in \mathbb{N})$
      **si** $b = 0$ **entonces**
          **regresa** $a$
      **si no**
          **regresa** Resta$(a - 1, b - 1)$

  El caso base es $b = 0$ y observamos que $a - b = a - 0 = a$ que es lo que devuelve el algoritmo. Sean $a, b$ los valores de la invocación original con $b > 0$, supondremos que la llamada *Resta*$(a', b')$ con $a' = a - 1, b' = b - 1, b' \geq 0$ es correcta. Nuestra función devuelve directamente lo que regresa la llamada recursiva. Observamos que $a - b = a' - b' = (a - 1) - (b - 1) = a - 1 - b + 1 = a - b$ que es lo que se deseaba demostrar.

- Demuestra que el siguiente algoritmo para calcular la resta de dos enteros no negativos es correcto:

  **función** $\textsc{Resta}(a, b \in \mathbb{N})$
      **mientras** $b \neq 0$
          $a \leftarrow a - 1$
          $b \leftarrow b - 1$
      **regresa** $a$

  El valor que se devuelve es la variable $a$ y el único lugar donde $a, b$ son modificados es dentro del ciclo. Propondremos la siguiente invariante: sean $a_0, b_0$ los valores originales de los argumentos, queremos que durante la $t$-ésima evaluación de la condición del ciclo (independientemente de si la condición fue verdadera o no), los valores de $a, b$ sean $a = a_0 - t, b = b_0 - t$. Verificamos a mano para $t = 0$ que $a = a_0 - 0 = a_0$ y $b = b_0 - 0 = b_0$, lo cual es cierto porque aún no se ha ejecutado ninguna iteración del ciclo. Supondremos que se sigue cumpliendo para $t = k$. Si la $t$-ésima evaluación de la condición fue falsa entonces $b = 0$ pero $b = b_0 - t \implies t = b_0$ y $a = a_0 - t \implies a = a_0 - b_0$ que es el resultado que queríamos calcular. Si la $t$-ésima evaluación de la condición fue verdadera entonces se ejecuta el ciclo y se evaluará la condición por $(t+1)$-ésima vez. Durante el ciclo, las variables pasan de $a = a_0 - t, b = b_0 - t$ hacia $a = a_0 - t - 1, b = b_0 - t - 1$. Como $a = a_0 - (t+1), b = b_0 - (t+1)$ se cumple la invariante al evaluar la condición por $(t+1)$-ésima vez.

- Dada la recurrencia $T(0) = 1$ y $T(n) = 1 + 2T(n-1)$ para toda $n > 0$, encuentra una $f$ tal que $T(n) \in \Theta(f)$.
  Podemos intentar evaluar $T$ para diferentes valores de $n$ en un intento por encontrar un patrón (la función $T$ crece muy rápido, por lo que hacer sustitución repetida nos dejará una gran cantidad de $1 + 1 + \cdots + 1$ y puede no ser claro lo que está pasando). Sabemos que $T(0) = 1$ y entonces $T(1) = 1 + 2T(0) = 3$, $T(2) = 1 + 2T(1) = 7$ y $T(3) = 1 + 2T(2) = 15$. Al parecer el patrón es $T(n) = 2^{n+1} - 1$ pero estrictamente tendríamos que demostrarlo. Para $T(0) = 1 = 2^{0+1} - 1 = 2 - 1 = 1$ y supondremos que $T(k) = 2^{k+1} - 1$. Tenemos que $T(k+1) = 1 + 2T(k) = 1 + 2(2^{k+1} - 1) = 1 + 2^{k+2} - 2 = 2^{k+2} - 1$ que es lo que se deseaba demostrar. Aunque ya sabemos que $T(n) = 2^{n+1} - 1$ y definir $f$ como tal haría que la demostración $T(n) \in \Theta(f)$ se volviera trivial, demostraremos que $T(n) \in \Theta(2^n)$.
  Es fácil demostrar que $T(n) \in O(2^n)$ si elegimos $c = 2, n_0 = 0$ porque entonces necesitamos demostrar que $2^{n+1} - 1 \leq 2(2^n)$ para toda $n \geq 0$, lo cual es obvio. Por otro lado, también debemos demostrar que $T(n) \in \Omega(2^n)$, lo cual se deja como ejercicio.