Este es el examen final del curso *Programación Imperativa Modular (PIMO)*, 2015-1. El examen tiene 7 preguntas, otorga un total de 50 puntos y 20 de bono. El examen es *individual* y no es permitido el uso de libros, apuntes ni equipos electrónicos.

Nombre y código: _____

| Pregunta | 1 | 2 | 3 | 4 | 5 | 6 | 7 | **Total** |
|---|---|---|---|---|---|---|---|---|
| Puntos | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 70 |
| Puntaje | | | | | | | | |

**Instrucciones:** Las preguntas están divididas en dos grupos: (i) preguntas 1 a 3 y (ii) preguntas 4 a 7. Para obtener 50 puntos es necesario que escoja 2 preguntas del grupo (i) y 3 preguntas del grupo (ii), indicando claramente su elección. Cualquier otra pregunta resuelta *completamente* será contabilizada como parte del bono.

1. (10 puntos) Considere la siguiente especificación:

   **Entrada:** un arreglo $a[0..N)$ de números, $N \geq 0$.

   **Salida:** el valor del producto $(\times i \mid 0 \leq i < N : a[i])$

   El siguiente fragmento de código Python implementa una solución del problema anterior:

```
1 def prod(a):
2   N = len(a)
3   r,n = 1,0
4   while n != N:
5     r,n = r*a[n],n+1
6   return r
```

   Considere los siguientes invariantes para el ciclo `while` en las líneas 4–5:

   $$P_0 : 0 \leq n \leq N \qquad\qquad P_1 : r = (\times i \mid 0 \leq i < n : a[i])$$

   Demuestre las propiedades de iniciación, estabilidad y terminación para $P_0$ y $P_1$. Concluya que, al momento de terminar, la función `prod(a)` calcula el producto de los números en `a`.

2. (10 puntos) Dados dos arreglos, por ejemplo $A[0..M)$ y $B[0..N)$, una *subsecuencia común* de $A$ y $B$ es una subsecuencia de $A$ que es también subsecuencia de $B$ (o viceversa). Considere el siguiente problema:

   **Entrada:** dos arreglos $A[0..M)$ y $B[0..N)$, con $M \geq 0$ y $N \geq 0$.

   **Salida:** el máximo entre las longitudes de las subsecuencias comunes de $A[0..M)$ y $B[0..N)$.

   También considere la siguiente función, con $0 \leq m \leq M$ y $0 \leq n \leq N$:

   $\phi(m,n)$ : "el máximo entre las longitudes de las subsecuencias comunes de $A[0..m)$ y $B[0..n)$."

   definida recurrentemente de la siguiente manera:

   $$\phi(m,n) = \begin{cases} 0 & \text{, si } m = 0 \vee n = 0 \\ 1 + \phi(m-1,n-1) & \text{, si } m > 0 \wedge n > 0 \wedge A[m-1] = B[n-1] \\ \phi(m,n-1) \uparrow \phi(m-1,n) & \text{, si } m > 0 \wedge n > 0 \wedge A[m-1] \neq B[n-1] \end{cases}$$

   Diseñe una tabulación para la función $\phi$ y un algoritmo que procese la tabulación de tal manera que la complejidad temporal del algoritmo sea $O(M \cdot N)$ y la espacial sea $O(M \uparrow N)$.

3. (10 puntos) Considere el tipo de datos `dlist` de listas doblemente encadenadas que se presenta a continuación:

```
class dlist:
  def __init__(self):
    self._sentinel = node()
```

```
class node:
  def __init__(self,prv=None,nxt=None,val=None):
    self._prev,self._next,self._value = prv,nxt,val
```

Internamente, `dlist` usa el tipo de datos `node` para representar los valores en la lista, además de un centinela de tipo `node`. Un objeto de tipo `node` referencia a otros objetos del mismo tipo (atributos `_prev` y `_next`) y tiene un valor (atributo `_value`). Con base en estas estructuras de datos y el desarrollo de *listas doblemente encadenadas con centinela* visto en clase, diseñe y documente las operaciones `insert_last(x)` y `remove_first()` para `dlist` que, respectivamente, inserta el valor `x` como último elemento de una lista y elimina el primer elemento de una lista en orden $O(1)$. Además, su diseño debe contemplar y manejar situaciones de error.

4. (10 puntos) Considere la especificación del siguiente problema:

**Entrada:** un grafo dirigido $G = (V, E)$ con $n \geq 0$ vértices $V = \{0, 1, \ldots, n-1\}$ y un vértice $v \in V$.

**Salida:** una lista ordenada ascendentemente con los vértices que no son alcanzables desde $v$.

Diseñe un algoritmo que resuelva el problema anterior y analice su complejidad algorítmica (i.e., temporal y espacial). Sea especialmente claro en la estructura de datos que utiliza para representar a $G$.

5. Considere un tablero de $r$ filas y $c$ columnas en donde la celda en la posición superior izquierda es identificada con la coordenada $(0,0)$ ya la posición inferior derecha es identificada con la coordenada $(r-1, c-1)$. El tablero puede ser atravezado por medio de saltos ortogonales entre celdas adyacentes (i.e., norte, oriente, sur y occidente). Considerando que cada una de las $r \times c$ celdas contiene un número natural (i.e., un número entero no negativo) que representa un costo, es razonable preguntar por el costo mínimo de un trayecto entre un par de celdas.
   (a) (2 puntos) Especifique el problema de determinar el costo mínimo de un trayecto desde una celda inicial hacia una celda final en un tablero de $r$ filas y $c$ columnas en donde cada celda contiene un número natural.
   (b) (8 puntos) Diseñe un algoritmo que resuelva el problema anterior y analice su complejidad algorítmica (i.e., temporal y espacial).

6. Una ciudad tiene $n$ intersecciones y $m$ caminos bidireccionales conectando pares de intersecciones. Cada camino tiene una capacidad de flujo vehicular medida en carros por minuto. Hay una ruta entre cualquier par de intersecciones, así esta esta no sea directa por medio de un camino. El departamento de mantenimiento de vías ha excedido su presupuesto operativo y necesita cerrar tantos caminos como sea posible sin *desconectar* la ciudad. Además, este cierre de vías quiere hacerce de tal manera que la capacidad mínima entre los caminos que queden sea *máxima*.
   (a) (2 puntos) Especifique el problema anterior.
   (b) (8 puntos) Diseñe un algoritmo que resuelva el problema especificado en el numeral anterior y analice su complejidad algorítmica (i.e., temporal y espacial).

7. Un grafo molecular es un concepto usado en química para representar la estructura de las moléculas. El *número de Wiener* de una molécula es usado para estudiar propiedades de los alcalinos. Suponga que una molécula es representada con un grafo conexo no dirigido $G = (V, E, w)$, con vértices $V = \{v_1, \ldots, v_n\}$ y función de peso $w(e) = 1$ para cualquier $e \in E$. El número de Wiener de $G$, denotado $\rho(G)$, se define como:

$$\rho(G) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} dist(v_i, v_j) = (+i, j \mid 1 \leq i < j \leq n : dist(v_i, v_j)),$$

en donde $dist(u, v)$ denota la distancia mínima entre los vértices $u$ y $v$ del grafo $G$.
   (a) (8 puntos) Diseñe un algoritmo que calcule $dist(u, v)$ para cualquier par de vértices $u$ y $v$ del grafo $G$, y cuya complejidad temporal sea $O(n^3)$.
   (b) (2 puntos) Usando como base la parte (a), diseñe un álgoritmo que determine $\rho(G)$.