Apellido y Nombre: ____________________

Carrera: ________ DNI: ____________________

[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Algoritmos y Estructuras de Datos. 1er Parcial. Tema: 1A. [21 de abril de 2005]

[Ej. 1] **[Clases (30 puntos)]** Escribir la implementación en `C++` del `TAD LISTA` (clase `list`) implementado por punteros **ó** cursores **ó** arreglos. Las funciones a implementar son `insert(p,x)`, `erase(p)`, `next()/iterator::operator++(int)`, `list()`, `clear()`. Observaciones:

- En caso de optar por escribir la interfase *"básica"*, debe escribir todas las declaraciones necesarias de la clase, tanto en la parte privada como pública.
- En caso de optar por la interfase *"avanzada"*, debe declarar e implementar completamente las partes privadas de la clase `list` e `iterator`.

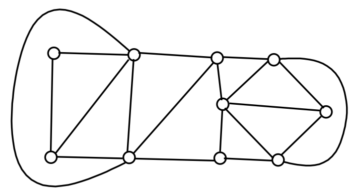
[Ej. 2] **[Programación (total = 50 puntos)]**

a) **[nilpot (25 puntos)]** Dadas dos correspondencias M_1 y M_2 la *"composición"* de ambas es la correspondencia $M = M_2 \circ M_1$ tal que si $M_1[a] = b$ y $M_2[b] = c$, entonces $M[a] = c$. Por ejemplo, si `M1={(0,1),(1,2),(2,0),(3,4),(4,3)}`, y `M2={(0,1),(1,0),(2,3),(3,4),(4,2)}`, entonces $M = M_1 \circ M_2$ `={(0,0),(1,3),(2,1),(3,2),(4,4)}`. Notemos que para que sea posible componer las dos correspondencias es necesario que los valores del contradominio de M_1 estén incluidos en las claves de M_2. Si el conjunto de valores del contradominio de una correspondencia M está incluido en el conjunto de sus claves, entonces podemos componer a M consigo misma, es decir $M^2 = M \circ M$. Por ejemplo, $M_1^2 = M_1 \circ M_1$`= {(0,2),(1,0),(2,1),(3,3),(4,4)}`. De la misma manera puede definirse, $M^3, ..., M^n$, componiendo sucesivamente. Puede demostrarse que, para algún n debe ser $M^n = I$, donde I es la *"correspondencia identidad"*, es decir aquella tal que $I[x] = x$. Por ejemplo, si `M = {(0,1),(1,2),(2,0)}`, entonces para $n = 3$, $M^n = M^3 = I$.
Consigna: Escribir una función `int nilpot(map<int,int> &M);` que dada una correspondencia `M` retorna el mínimo entero `n` tal que $M^n = I$.
Sugerencia: Escribir dos funciones auxiliares:

- `void compose(map<int,int> &M1,map<int,int> &M2,map<int,int> &M);` que dadas dos correspondencias `M1, M2`, calcula la composición $M = M_2 \circ M_1$, devolviéndola en el argumento `M`,
- `bool is_identity(map<int,int> &M);` que dada una correspondencia `M`, retorna `true` si `M` es la identidad, y `false` en caso contrario.

b) **[chunk-revert (25 puntos)]** Escribir una función `void chunk_revert(list<int> &L,int n);` que dada una lista `L` y un entero `n`, invierte los elementos de la lista tomados de a `n`. Si la longitud de la lista no es múltiplo de `n` entonces se invierte el resto también. Por ejemplo, si `L={1,3,2,5,4,6,2,7}` entonces después de hacer `chunk_revert(L,3)` debe quedar `L={2,3,1,6,4,5,7,2}`. *Restricciones:* Usar a lo sumo una estructura auxiliar. (En tal caso debe ser lista, pila o cola).

[Ej. 3] **[color-grafo (5 ptos)]** Colorear el grafo de la figura usando el mínimo número de colores posible. Usar el algoritmo heurístico ávido. ¿La coloración obtenida es óptima? Justifique.



[Ej. 4] **[Preguntas (total = 15 puntos, 5puntos por pregunta)]** Responder según el sistema *"multiple choice"*, es decir marcar con una cruz el casillero apropiado. **Atención:** Algunas respuestas son intencionalmente "descabelladas" y tienen puntajes **negativos**!!]

a) Considere la función:

```
bool is_mapped(map<int,int> &M,int key,int val) {
  return /* Esta key -> val en M? ... */; }
```

que debe retornar `true` si el par de asignación `(key,val)` está en la correspondencia `M`. ¿Cuál es la expresión correcta que refina el seudocódigo?

- ☐ ... `M[key]==val`
- ☐ ... `M[key]==val && (M.find(key)!=M.end())`
- ☐ ... `M.find(key)->second==val`
- ☐ ... `(M.find(key)!=M.end()) && M[key]==val`

---

b) El tiempo de ejecución de la función `find(key)` para correspondencias implementadas por vectores ordenados es ... (n es el número de asignaciones en la correspondencia)

- ☐ ... $O(1)$
- ☐ ... $O(\log n)$
- ☐ ... $O(n)$
- ☐ ... $O(n^2)$

---

c) Dadas las funciones

- $T_1(n) = 3n^3 + 2n! + \log n$,
- $T_2(n) = 3 \cdot 2^3 + n^2 + n^{1.5}$,
- $T_3(n) = 5! + 6 \cdot 2^n + 20 \cdot n^2$ y
- $T_4(n) = 2^{10} + 20n + \log_2 40$

ordenarlas de menor a mayor.

$$T_{\square} < T_{\square} < T_{\square} < T_{\square}$$