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

Algoritmos y Estructuras de Datos. 1er Parcial. Tema: 1A. [20 de abril de 2006]

[Ej. 1] [Clases (30 puntos)]

a) 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)`. 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 completamente las partes privadas de la clase `list`. **NO** es necesario implementar las clases `cell` ni `iterator`.

b) Escribir la implementación en `C++` del `TAD PILA` o del `TAD COLA` (a elección).

[Ej. 2] [Programación (total = 45 puntos)]

a) **[ascendente (15 puntos)]**
En ciertas aplicaciones interesa separar las corridas ascendentes en una lista de números $L = (\ a_1,\ a_2,\ ...,\ a_n\ )$, donde cada corrida ascendente es una sublista de números consecutivos a_i, a_{i+1}, ..., a_{i+k}, la cual termina cuando $a_{i+k} > a_{i+k+1}$, y es ascendente en el sentido que $a_i \leq a_{i+1} \leq ... \leq a_{i+k}$. Por ejemplo, si $n = 10$ y la lista es L = (0,5,6,9,4,3,9,6,5,5,2,3,7), entonces hay 6 corridas ascendentes, a saber: (0,5,6,9), (4), (3,9), (6), (5,5) y (2,3,7).
Consigna: usando las operaciones de la clase lista, escribir una función `int ascendente(list<int> &L,list<list<int> > &LL)` en la cual, dada una lista de enteros `L`, almacena cada corrida ascendente como una sublista en la lista de listas `LL`, devolviendo además el número z de corridas ascendentes halladas. *Restricciones:* a) El tiempo de ejecución del algoritmo debe ser $O(n)$, b) La lista de listas `LL` inicialmente está vacía, c) No usar otras estructuras auxiliares.

b) **[apply-map (15 puntos)]** Escribir una funcion `void apply_map(list<int> &L, map<int,int> &M, list<int> &ML)` que, dada una lista `L` y una correspondencia `M` retorna por `ML` una lista con los resultados de aplicar `M` a los elementos de `L`. Si algún elemento de `L` no está en el dominio de `M` entonces el elemento correspondiente de `ML` no es incluido. Por ejemplo, si

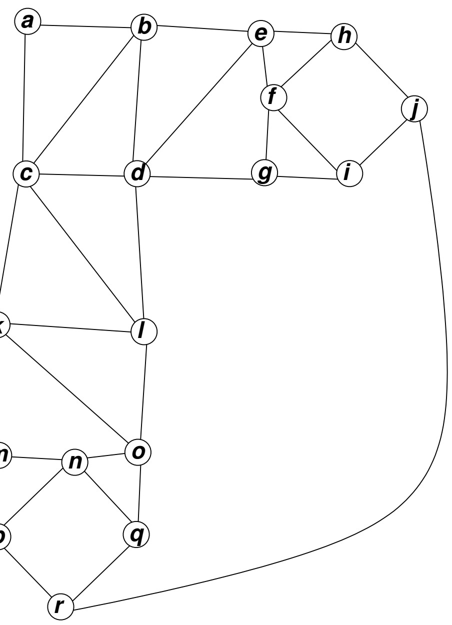
$$\mathtt{L} = (1,2,3,4,5,6,7,1,2,3)$$
$$\mathtt{M} = \{(1,2),(2,3),(3,4),(4,5),(7,8)\}$$

entonces después de hacer `apply_map(L,M,ML)`, debe quedar

$$\mathtt{ML} = (2,3,4,5,8,2,3,4)$$

Restricciones: No usar estructuras auxiliares. El tiempo de ejecución del algoritmo debe ser $O(n)$, donde n es el número de elementos en la lista (asumiendo que las operaciones usadas de correspondencia son $O(1)$).

c) **[inverse (15 puntos)]** Dada una correspondencia `M` y asumiendo que es *invertible* o *biunívoca* (esto es, todos los valores del contradominio son distintos), la correspondencia *"inversa"* `N` es aquella tal que, si `y=M[x]`, entonces `x=N[y]`. Por ejemplo, si `M={(0,1),(1,2),(2,0)}`, entonces la inversa es `N={(1,0),(2,1,(0,2))}`. *Consigna:* Escribir una función `bool inverse(map<int,int> &M,map<int,int> &N)` tal que, si `M` es invertible, entonces retorna true y `N` es su inversa. En caso contrario retorna falso y `N` es la correspondencia *"vacía"* (sin asignaciones).



[Ej. 3] **[color-grafo (5 ptos)]** Colorear el grafo de la figura usando el mínimo número de colores posible. Usar el algoritmo heurístico partiendo del vértice a y recorriendo los vértices en orden alfabético $b, c, d, \ldots$. ¿La coloración obtenida es óptima? Justifique.

[Ej. 4] **[Preguntas (total = 20 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) ¿Cuál es el tiempo de ejecución de `x=M[key]` para correspondencias implementadas con vectores ordenados?

- ☐ ... $O(n)$
- ☐ ... $O(\sqrt{n})$
- ☐ ... $O(1)$
- ☐ ... $O(\log n)$

---

b) ¿Cuál es el tiempo de ejecución para `L.insert(p,x)` en listas implementadas por arreglos?

- ☐ ... $O(\sqrt{n})$
- ☐ ... $O(\log n)$
- ☐ ... $O(1)$
- ☐ ... $O(n)$

---

c) Dado el siguiente fragmento de código

```
map<int,int>M;
M[0]=1;
int i = M[3];
```

¿Cuál de las siguientes aseveraciones es verdadera?

- ☐ ... `M` no se modifica y devuelve `i=3`.
- ☐ ... `M` se modifica y devuelve `i=0`.
- ☐ ... Da un error.
- ☐ ... `M` se modifica y devuelve `i=1`.

---

d) Dadas las funciones

- $T_1(n) = 2n^2 + 1000\,n + \sqrt{5n} + 2^{20}$,
- $T_2(n) = n! + 2n^5 + 5n^2 + 3n^{5/2}$,
- $T_3(n) = 3n^2 + 2n^3 + n^{3/2} + 3\,\log n$,
- $T_4(n) = 4n^{1.5} + 3n^{2.5} + n\log n + 6\sqrt{n}$.

ordenarlas de menor a mayor.

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