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.
Examen Final. [14 de Febrero de 2008]

[Ej. 1] [Clases (mínimo 60 %)]

a) Escribir las funciones de los TAD cola y pila (clases `queue<>` y `stack<>`). Incluir el header y todas las declaraciones necesarias.

b) Escribir la función de ordenamiento por fusión `void merge_sort(list<int> &L)` para listas de enteros. Los enteros se comparan por el operador `<`.

[Ej. 2] [Programación (mínimo 40 %)]

a) [**verifica-abb**] Escribir una función predicado
`bool verifica_abb(tree<int> &A,bool (*less)(int,int));` que verifica si el subárbol de un nodo `n` verifica la condición de *árbol binario de búsqueda* con respecto a la relación de orden `less()`.

b) [**elimina-valor**]
Escribir una función `void elimina_valor(queue<int>&C, int);` que elimina todas las ocurrencias del valor `n` en la cola `C`. Por ejemplo, si `C = {1,3,5,4,2,3,7,3,5}`, después de `elimina_valor(C,3)` debe quedar `C = {1,5,4,2,7,5}`. *Sugerencia: Usar una estructura auxiliar lista o cola.*
Restricciones: El algoritmo debe tener un tiempo de ejecución $O(n)$, donde n es el número de elementos en la cola original.

[Ej. 3] [operativos (mínimo 70 %)]

a) [**heap-sort**] Dados los enteros $\{6, 8, 11, 4, 5, 9, 6, 7, 10\}$ ordenarlos por el método de *"montículos"* (*"heap-sort"*). Mostrar el montículo (minimal) antes y después de **cada** inserción/supresión.

b) [**huffman**] Dados los caracteres siguientes con sus correspondientes probabilidades, construir el código binario y encodar la palabra `KATHRINA` $P(K) = 0{,}05, P(H) = 0{,}2, P(T) = 0{,}1, P(R) = 0{,}05, P(A) = 0{,}2, P(N) = 0{,}15, P(I) = 0{,}1, P(P) = 0{,}05, P(S) = 0{,}05, P(B) = 0{,}05$. Calcular la longitud promedio del código obtenido.

[Ej. 4] [Preguntas (mínimo 70 %)]

a) ¿Puede tener una correspondencia varios valores iguales del dominio, o sea varias claves iguales? (Por ejemplo `M1={(1,2),(1,3)}`) ¿Y varios valores iguales del contradominio? (Por ejemplo `M2={(1,2),(3,2)}`)

b) Que retorna la dereferenciación de un iterator sobre correspondencias (clase `map`).

c) ¿Qué ocurre si se invoca el `operator[]` sobre una correspondencia con una clave que no tiene asignación. Por ejemplo `M={(1,2),(3,4)}` y hacemos `x = M[5]`. ¿Da un error? ¿Qué valor toma `x`?

d) ¿Cuál es la condición de Arbol Binario de Búsqueda?

e) ¿Cómo se encuentra el mínimo y el máximo de los valores en un árbol binario de búsqueda? ¿Cuál es la complejidad algorítmica de esta operación (caso promedio, mejor y peor)?

f) ¿Porqué decimos que $(n+1)^2 = O(n^2)$ si siempre es $(n+1)^2 > n^2$?

g) Discuta si es posible insertar en una posición dereferenciable en árboles binarios.

h) Discuta el valor de retorno del método `pair<set::iterator,bool> set::insert(T x)`.

i) Defina árbol binario completo y árbol binario lleno. ¿Es verdad que todo árbol binario completo es lleno? ¿Y viceversa?

j) ¿Cuál es el número de intercambios (en notación $O()$) que requieren los algoritmos de ordenamiento lentos en el peor caso?