Universidad Nacional de Río Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Departamento de Computación
Estructura de Datos yAlgoritmos – Algoritmos I
Septiembre de 2001

Trabajo Práctico Nº 3 : Tipo Abstracto de Datos CONJUNTO – DICCIONARIO - ESTRUCTURAS MÚLTIPLES

Ej 1) Resuelva los ejercicios pendientes en el tórico.

Ej 2) Implemente el TAD Diccionario usando:

a. arreglos con tope.
b. TAD LISTA (impl.con punteros)
c. árboles binarios de búsqueda

Ej 3) Implemente el TAD Set:

a. listas encadenadas Ordenadas
b. Arreglo de bits.

Ej 4) Extienda la clase implementada en el ejercicio anterior, sobrecargando el operador == para objetos derivados de esta clase y pruebe las siguientes igualdades.

a- Suprimir (Insertar (A , x) , x) = A
b- Insertar (Suprimir (A , x) , x) = A
c- Union (A , {}) = A
d- Interseccion (A , B) = B
e- Insertar (Diferencia (A , B) , x) = Union (A , {x})

Ej. 5) Se desea llevar cierta información a cerca de las casas de una ciudad.
Para ello debera informatizar una pequeña administración de dichas casas.
Los atributos de una casa son:

- DNI del Propietario
- Cantidad de ambientes
- Barrio
- Cotización.

Definir el TDA Casas, donde las operaciones a efectuar sobre dichas casas son:

- Insertar una casa.
- Eliminar una casa.
- Mostrar por pantallas las casas de un cierto **Barrio**.
- Dados 2 **Barrios** imprimir aquellos propietarios que tienen casas en ambos barrios.

Ej 6) Implemente el TAD Diccionario de **datos** utilizando la técnica de *hashing abierto*. el tipo **datos** es el siguiente (la clave aquí es el nombre).

```
tipo datos = registro
                nombre: cadena de 20 caracteres
                edad: natural entre 0 y 150
                estado civil: (soltero, casado, viudo)
             fin.
```

utilice la siguiente función de *Hashing:*

```
funcuón h(dato c:cadena de 20 caracteres)-> natural entre 0 y 26
```

Ej 7) Implemente los ejemplos vistos en el teórico.

Ej 8) Considere cola de prioridad de enteros con las siguientes operaciones:

```
Crear       : P(int) → cola de prioridad
Insertar    : cola de prioridad × int → cola de prioridad
```

```
Minimo      : cola de prioridad → int
BorrarMin   : cola de prioridad → cola de prioridad
```

En ciertas aplicaciones de colas de prioridad, por ejemplo administración de tareas en un sistema operativo, se desea poder aumentar y disminuir la prioridad de un elemento, asi como también borrar tareas arbitrarias. Para ello se extiende el TAD cola de prioridad con operaciones decrementar llave; incrementar llave y eliminar. Donde decrementar llave(x; d; M) reduce el valor de la llave en la posición x por una cantidad positiva d e incrementar llave(x; d; M) aumenta el valor de la llave en la posición x en una cantidad positiva d. Para eliminar un elemento, se usa la operación decrementar llave(x; 1; M) y luego borrarmin. Para que la extención del TAD con estas operaciones sea efectiva, se debe contar con una estructura adicional que permita dado un valor, determinar cual es su posición dentro del heap. Diseñe estructuras de datos apropiadas que permitan extender el TAD cola de prioridad extendida (cpe) con estas operaciones. Implemente el TAD cola de prioridad extendido con operaciones:

```
Crear            : P(int) → cpe
Insertar         : cpe × int → cpe
Minimo           : cpe → int
BorrarMin        : cpe → cpe
DecrementarLlave: cpe × int → cpe
IncrementarLlave: cpe × int → cpe
Eliminar         : cpe × int → cpe
```

Ej 9) Un Departamento de Computación que se creó hace 30 años, decidió implementar un sistema de almacenamiento y recuperación de información relacionado con las tesis de doctorado que otorgó a lo largo de su historia. Dentro de cada año, las tesis se enumeran consecutivamente empezando de 1, y hay un máximo de 30 y un promedio de 20 tesis otorgadas en un año. Asi, por ejemplo, la tesis con número 1984/7 es la séptima tesis defendida en el año 1984.
Las tesis contienen información sobre el nombre del autor, el nombre del supervisor, el título de la misma, el año y el número de tesis dentro del año. Un autor solo puede defender una tesis de doctorado en su carrera. Se puede asumir que no hay ni dos autores ni dos supervisores con el mismo nombre.

Se desea satisfacer las siguientes operaciones:

- **Tesis SuTesis(Nombre autor, Historia historial)**, en el que dados el autor de una tesis y el historial con todas las tesis otorgadas por el departamento, devuelve la información relacionada con la tesis defendida por dicho autor. Esta operación se debe realizar en O(1) promedio.
- **LasTesis QueTesis (int anho)**, en el que dado un año, devuelve todas las tesis defendidas dicho año, ordenadas por número. Esta operación se debe realizar en O(n) peor caso, donde n es la cantidad de tesis defendida dicho año.
- **LasTesis Supervisadas(Nombre supervisor)**, en el que dado el nombre de un supervisor, devuelve todas las tesis que orientó, ordenadas por año, y dentro de año, por número. Esta operación se debe realizar en O(n) peor caso, donde n es la cantidad de tesis que orientó.
- **void AltaTesis(Nombre autor, Nombre supervisor, string titulo, int anho, int numero, Historia *historial)**, en el que dados los nombres del autor y supervisor de la tesis, su título, el año en que fue defendida, su enumeración dentro del año, y el historial, agrega la información sobre dicha tesis en el historial. Esta operación se debe realizar en O(1) promedio. Se puede asumir que las tesis son dadas de alta en el mismo orden cronológico en la cual fueron defendidas, y que la información sobre el supervisor ya está disponible en el historial.

Se pide:

a) Diseñar estructuras de datos apropiadas para implementar eficientemente dichas operaciones, describiendo como se obtienen las cotas de tiempo pedidas.
b) Dar una declaración en C++ de los tipos de datos descritos en la parte a).
c) Escribir en C++ la función AltaTesis.