UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y CC. Computación
ETSI Informática

Laboratorio de Programación III

Examen Septiembre 99

Alumno: _____

Grupo 2º _____ Especialidad _____ Odenador: _____

1. Implementar el TAD lista posicional con cota variable. Este tipo se diferencia de una lista posicional no acotada en que en un instante dado, sólo puede haber un número limitado de elementos en la lista, pero este límite se puede variar durante la utilización de la lista. Este límite puede tomar cualquier valor.
El tipo de la lista se define como:

```
TYPE
    Lista_Enlazada =    POINTER TO Nodo;
    Nodo =              RECORD
                            x : ITEM;
                            sig : Lista_Enlazada;
                        END;
    T_ListaAc =         POINTER TO Cabecera;
    Cabecera =          RECORD
                            cota: CARDINAL;
                            longitud: CARDINAL;
                            datos: Lista_Enlazada;
                        END;
```

El módulo de definición debe ser el siguiente:

```
DEFINITION MODULE ListaAc;
    TYPE  T_ListaAc;
          ITEM=     <<a definir >>
          ERROR=    <<a definir >>

    PROCEDURE Error():ERROR;

(* Crea una lista vacía. El número máximo de elementos para esa lista viene dado por cota *)
    PROCEDURE Crear(VAR l:T_ListaAc; cota: CARDINAL);

(* Devuelve el número de elementos que hay en la lista l. *)
    PROCEDURE Longitud(l: T_ListaAc): CARDINAL;

(* Cambia el número máximo de elementos de la lista l a cota. Si la nueva cota es menor que el
número de elementos actual de la lista, se borran los últimos hasta llegar a la cota.*)
    PROCEDURE CambiarCota(VAR l: T_ListaAc; cota: CARDINAL);

(* Devuelve el número máximo de elementos que se puede almacenar en la lista l *)
    PROCEDURE VerCota(l: T_ListaAc): CARDINAL;
    END ListaAc.

 (* Devuelve VERDADERO si no hay ningún elemento en la lista. FALSO en caso contrario *)
    PROCEDURE Vacia(l: T_ListaAc):BOOLEAN;
```

```
(* Inserta el elemento x en la posición pos de la lista l. Los elementos que están a partir de la
posición pos pasan a la siguiente posición. Si la lista está llena, no se inserta el elemento. *)
    PROCEDURE Insertar(VAR l: T_ListaAc;x:ITEM;pos:CARDINAL);

(* Elimina el elemento de la posición pos de la lista l. Los elementos que están a partir de la
posición pos pasan a la posición anterior. *)
    PROCEDURE Eliminar(VAR l:T_ListaAc;pos:CARDINAL);

(* Devuelve en la variable x el valor de la posición pos de la lista l. *)
    PROCEDURE Elemento(l:T_ListaAc;pos:CARDINAL; VAR x: ITEM);

(* Libera la memoria ocupada por la lista l. *)
    PROCEDURE Destruir(VAR l: T_ListaAc);
```

2. Haciendo uso del TAD anterior implementar una tabla de dispersión. Esta tabla guardará nombres de personas junto con una clave identificativa (un natural en el rango [1 .. MaxValor]) que debe ser única. Para resolver las colisiones se usará una estrategia *cubo*. Por cada entrada de la tabla se tiene una lista acotada que permite almacenar hasta un máximo de elementos que hayan colisionado. Si se produce otra colisión para ese valor de la función de dispersión se debe resolver con una función de *rehash*.

```
    DEFINITION MODULE Tabla_D;
    TYPE T_Disp;
        NOMBRE =        <<a definir>>;
        T_Clave =       <<a definir>>;
        ITEM =          <<a definir>>;
        ERROR =         <<a definir>>;
        COMPARAR =      <<a definir>>;
    PROCEDURE Error():ERROR;

(* Este procedimiento crea una tabla de dispersión vacía *)
    PROCEDURE Crear(): T_Disp;

(* Este procedimiento inserta un elemento en la tabla de dispersión. *)
    PROCEDURE Insertar(VAR t_d: T_Disp; x: ITEM; comp: COMPARAR);

(* Este procedimiento elimina de la tabla el elemento con la clave cl. *)
    PROCEDURE Extraer(VAR t_d: T_Disp ; cl:T_Clave; comp: COMPARAR);

(* Este procedimiento devuelve en la variable x el valor del elemento cuya clave es cl. *)
    PROCEDURE Elemento(t_d: T_Disp ; cl:T_Clave; comp:COMPARAR;
                                    VAR x:ITEM);
    END Tabla_D.
```

Nota: En la implementación de la tabla de dispersión no se puede usar el procedimiento *CambiarCota* de la lista posicional.

3. Construir un programa de prueba para verificar el correcto funcionamiento de la tabla de dispersión.