PRÁCTICA 3

Tipo Abstracto de Datos: Listas simplemente enlazadas, listas doblemente enlazadas, listas circulares. Nota: para implementar los TADs utilice archivos .c y .h

IMPORTANTE: para cada función pedida en los TADs analizar, diseñar y programar todos los casos:

- **Si la operación se puede llevar a cabo, determinar qué retorna la función y el estado de las distintas variables y / o parámetros.**
- **Casos en que la operación no se puede llevar a cabo: a modo de ejemplo, eliminar el nodo de la posición 10 siendo la longitud de la lista 5. Determinar precondiciones y postcondiciones, decidir qué hace la función en estos casos, programar y documentar. Las funciones deben funcionar correctamente para todos los casos.**
- **Analizar y justificar los parámetros de las funciones. Detectar para cada parámetro si está pasado por valor o por referencia y justificar el por qué.**

1. LISTAS

1) Implementar el TAD lista de enteros utilizando punteros. Una posible implementación del tipo sería:

```
struct Nodo
{
  int dato;
  struct Nodo *sig;
};

typedef struct
{
  int n;
  struct Nodo *lista;
}Lista_T;
```

El TAD debe disponer de las siguientes funciones:

1. `Lista_T crearLista();`
   // crea una lista del tipo `Lista_T` con 0 elementos y la retorna (lista vacía).
2. `int InsertarPrimero(Lista_T *l, int x );`
   // inserta nodo con elemento x como primer elemento de la lista. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

3. `int InsertarUltimo(Lista_T *l, int x );`
   // inserta nodo con elemento x como último elemento de la lista. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

4. `int RecorrerLista(Lista_T l );`
   // Recorre la lista "l" imprimiendo sus elementos. Retorna 1 si la operación se realiza con éxito, -1 caso contrario (la lista no tiene elementos).

5. `int EstaVacia(Lista_T l);`
   // Retorna TRUE (1) si la lista está vacía, FALSE (0) caso contrario.

6. `int VaciarLista(Lista_T *l);`
   // Desaloca la memoria de cada nodo. Longitud de la lista = 0. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

7. `int SuprimirDato(Lista_T *l, int x);`
   // Elimina el nodo con el dato x de la lista. Determinar la acción a realizar en caso que el dato no exista en la lista y determinar los valores retornados.

8. `int SuprimirNodo(Lista_T *l, int pos);`
   // Elimina el nodo con posición `pos` de la lista. Determinar la acción a realizar en caso que la posición no exista en la lista y determinar los valores retornados.

9. `int LongitudLista(Lista_T l);`
   // Retorna la longitud de la lista.

10. `int DevolverDatoPosicion(Lista_T l, int pos);`
    // Devuelve el dato de la posición `pos`. Determinar la acción a realizar en caso que la posición no exista en la lista y determinar los valores retornados.

2) ¿Como implementaría el TAD lista de enteros con arreglos? Para cada operación implementada en 1) explique cómo lo implementaría con arreglos. Compare ambas implementaciones desde el punto de vista de memoria utilizada. Para cada uno, defina si la operación es más simple o más costosa a nivel de cantidad de operaciones que debe realizar (comparaciones, corrimientos, etc).

3) Responda:

(a) ¿Cuál de las 2 implementaciones anteriores es más conveniente? ¿Qué aspectos tendría en cuenta para optar por una de las dos implementaciones?

(b) Cuál de las 2 implementaciones elije si:

(I) No se sabe la cantidad de elementos de la lista. No es posible acotar una posible longitud de la lista.

(II) Se realizan muchas operaciones `InsertarPrimero()`, `InsertarUltimo()` y `SuprimirDato()`.

(III) Sobre todo, se realizan operaciones `devolverDatoPosicion()` sobre la lista.

4) Se desea implementar una lista de enteros. Se pide obtener la cantidad de espacio de memoria utilizado para las implementaciones:

1. La lista tiene 100 elementos y se implementa con punteros.

2. La lista tiene 100 elementos y se tiene como precondición que la cota superior de elementos de la lista es de 100 elementos. Se implementa con arreglos.

Indique ventajas y desventajas de cada implementación. Indique qué implementación utilizaría y justifique su respuesta.

5) Utilizando el TAD implementado en 1) escriba un programa que agregue los datos 4, 5, 6 al comienzo, 7, 8 y 9 al final y que agregue en distintas posiciones los elementos 11 y 12. Luego, elimine los elementos pares de la lista.

6) ¿Cómo haría para recorrer la lista en orden inverso?

7) Se desea formar una lista enlazada de números aleatorios. El programa que realiza esta tarea inserta los nuevos nodos por la cabeza de la lista. Una vez creada la lista, se recorren los nodos para mostrar los números pares. Implementar con funciones.

9) Un conjunto es una colección de elementos del mismo tipo sin duplicidades. Escribir un programa para representar un conjunto de enteros mediante una lista enlazada. El programa debe contemplar las operaciones:
- Cardinal del conjunto.
- Pertenencia de un elemento al conjunto.
- Añadir un elemento al conjunto.
- Escribir en pantalla los elementos del conjunto.

10) El TAD Lista Ordenada, es similar a la lista simple, pero cada elemento que se agrega a la lista se agrega de forma ordenada, según un criterio de orden. Este criterio puede ser un campo clave o cualquier otra comparación de sus elementos. Cada vez que se agrega un dato a la lista ordenada, se busca su posición correcta y se inserta. Es un post condición que luego de insertar un dato a una lista ordenada, la misma sigue siendo una lista ordenada. Implementar este TAD para elementos del tipo complejo donde se pide que queden ordenados por su magnitud (magnitud de z = $\sqrt{(re(z)^2 + im(z)^2}$).

2. LISTAS DOBLEMENTE ENLAZADAS

11) Implementar el TAD lista doblemente enlazada de enteros utilizando punteros. Una posible implementación del tipo sería:

```
struct NodoDE
{
  int dato;
  struct NodoDE *sig;
  struct NodoDE *ant;
};

typedef struct
{
  int n;
  struct Nodo *primero;
  struct Nodo *ultimo;
}ListaDE_T;
```

El TAD debe disponer de las siguientes funciones:

1. `ListaDE_T crearLista();`
   // crea una lista del tipo `ListaDE_T` con 0 elementos y la retorna.

2. `int InsertarPrimero(ListaDE_T *l, int x );`
   // inserta nodo con elemento x como primer elemento de la lista. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

3. `int InsertarUltimo(ListaDE_T *l, int x );`
   // inserta nodo con elemento x como último elemento de la lista. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

4. `int RecorrerListaEnOrden(ListaDE_T l );`
   // Recorre la lista l imprimiendo sus elementos, del primer elemento al último. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

5. `int RecorrerListaEnOrdenInverso(ListaDE_T l);`
   // Recorre la lista l imprimiendo sus elementos, del último elemento al primero. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.

6. `int EstaVacia(ListaDE_T l);`
   // retorna True (1) si la lista está vacía, False (0) caso contrario.

7. `int SuprimirDato(ListaDE_T *l, int x);`
   // Elimina el nodo con el dato x de la lista. Determinar qué hacer si el dato no está en la lista. Determinar valores a retornar por la función.

8. `int VaciarLista(ListaDE_T *l);`
   // desaloca la memoria de cada nodo. Longitud de la lista = 0.

9. `int LongitudLista(ListaDE_T l);`
   // retorna la longitud de la lista

10. `int DevolverDatoPosicion(ListaDE_T l, int pos);`
    // devuelve el dato de la posición pos. Determinar que sucede si la posición no es válida. Determinar valores retornados.

11) Realizar un programa que inserte en una lista los enteros: 4, 3, 7, 10, 11 y 12. Luego, imprima la lista de dos formas: en el orden que tiene la lista y en el orden inverso. Agregar los datos 45, 67, 87 y luego eliminar los datos 11, 12 y mostrar la lista en orden inverso.

12) Realizar un programa que mantenga y manipule información sobre una encuesta relacionada al COVID-19. Las encuestas son anónimas y los datos que se almacenan de cada persona son: edad y temperatura corporal. Se pide mantener los datos ordenados por edad. Luego se pide imprimir los datos de menor a mayor edad imprimiento también sus temperaturas. Además se pide informar la cantidad de personas con edad mayor a 65 que tengan temperatura mayor a 39 grados, pero esta vez los datos deben ser impresos de mayor edad a menor edad. (Ayuda: armar la lista en orden, esto es, en cada inserción el dato debe quedar en su posición correcta a fin de mantener el orden).

3. Listas Circulares

12) Implementar el TAD lista circular con punteros. Una posible implementación del tipo sería:

```
struct NodoT {
   int dato;
   struct NodoT *sig;
};
typedef struct NodoT Nodo;

typedef struct{
   int n;
   Nodo *lista;
}ListaC_T;
```

El TAD debe disponer de las siguientes funciones:

1. `ListaC_T CrearLista();`
   // crea una lista circular con 0 elementos y la retorna.
2. `int InsertarDato(ListaC_T *l, int x );`
   // inserta nodo con el elemento x al comienzo de la lista. Retorna 1 si la operación se realiza con éxito, -1 caso contrario.
3. `int ImprimirLista(ListaC_T l);`
   // imprime la lista, sin importar desde donde.
4. `int LongitudLista(ListaC_T l );`
   // Retorna la longitud de la lista l.

13) Utilizando el TAD lista circular, implemente un programa que genere una lista con los datos 1, 2, 3, 4, 5 y 6 y luego imprima todos los datos de la lista.