UNIVERSIDAD CATÓLICA ANDRÉS BELLO – EXTENSIÓN GUAYANA

Escuela de Ingeniería Informática

Manual de Prácticas: Algoritmos y Estructuras de Datos

Actividad 4

Listas Simplemente Enlazadas

Parte 1: Listas enlazadas

Dada la siguiente definición de una lista enlazada:

```
typedef struct node {
    int value;
    struct node *next;
} Node;
```

Se desea que usted implemente las siguientes funciones utilizando dicha estructura.

Ejercicio 1: Funciones básicas de listas enlazadas

Implemente las siguientes funciones sobre listas enlazadas en su versión **recursiva**:

1. **add_end**: añade un nuevo elemento al final de la lista.

   ```
   Node *add_end(Node *listp, Node *newp);
   ```

2. **insert**: inserta un elemento en la lista en orden ascendente de acuerdo a su valor.

   ```
   Node *insert(Node *listp, Node *newp);
   ```

3. **delete_item**: elimina el primer nodo cuyo valor coincida con el valor especificado. La función busca el valor en la lista y, al encontrarlo, lo elimina y ajusta los punteros para mantener la estructura de la lista.

   ```
   Node *delete_item(Node *listp, int value);
   ```

4. **search**: busca un elemento en la lista de forma recursiva. La función retorna un apuntador al nodo que contiene el valor, o `NULL` si el valor no se encuentra en la lista.

   ```
   Node *search(Node *listp, int value);
   ```

5. **length**: calcula la longitud de una lista enlazada. La función retorna el número de nodos en la lista.

   ```
   int length(Node *listp);
   ```

6. **free_all**: libera todos los nodos de una lista, asegurando que no quede memoria sin liberar al final de la ejecución. La función no retorna ningún valor.

   ```
   void free_all(Node *listp);
   ```

7. **print**: muestra los elementos en la lista enlazada. La función no retorna ningún valor.

   ```
   void print(Node *listp);
   ```

8. **print_reverse**: imprime los elementos de la lista en orden inverso, sin modificar ni invertir la lista.

   ```
   void print_reverse(Node *listp);
   ```

Ejercicio 2: Funciones adicionales de listas enlazadas

A partir de las funciones vistas en clase, se desea que usted implemente las siguientes funciones, tanto en su versión **iterativa** como en su versión **recursiva**:

1. **reverse**: invierte una lista enlazada donde `listp` apunta al primer elemento. Retorna un apuntador al primer elemento de la lista invertida.

   ```
   Node *reverse(Node *listp);
   ```

2. **copy**: copia una lista enlazada donde `listp` apunta al primer elemento de la lista original, y retorna un apuntador al primer elemento de la nueva lista.

   ```
   Node *copy(Node *listp);
   ```

3. **concat**: concatena dos listas enlazadas dadas. Dada una lista `list1` y otra lista `list2`, la función retorna un apuntador a la nueva lista que contiene los elementos de ambas listas concatenadas.

   ```
   Node *concat(Node *list1, Node *list2);
   ```

4. **sort**: ordena una lista enlazada de manera ascendente donde `listp` apunta al primer elemento. La función retorna un apuntador al primer elemento de la lista ordenada.

   ```
   Node *sort(Node *listp);
   ```

Parte 2: Enteros grandes

Ejercicio 3: Enteros grandes con listas enlazadas

Defina el tipo **EnteroGrande** para representar enteros de tamaño arbitrario usando listas enlazadas.

Implemente la función para multiplicar dos enteros grandes. Se debe garantizar que la operación funcione correctamente sin limitaciones de tamaño de los enteros.