Linked List Variations

Circular Linked

In this version, the next pointer of the last node points to the first node of the list.

How will we detect the end of the list now?

Answer: If a node's next pointer points to the front of the list, then that node is the last in the list.

Most everything we must do for a circular linked list mirrors what we do for a regular linked list. The only difference is to maintain a link from the last node of the list to the first.

Consider inserting to the front of a circular linked list:

1) Create the new node to add.

2) If the original list is null, set the next to point to the new node itself and return a pointer to this node.

3) Otherwise, set its next pointer to the first node in the original list.

4) Iterate to the last node in the original list.

5) Change its next pointer to point to the newly added node.

6) Designate the pointer to the newly added node to be the new

front of the list.

In a similar manner, you should be able to come up with the steps necessary to insert to the back, in order, or to delete a node from a circular linked list.

Doubly Linked List

The data structure for this changes as follows:

```
struct dll {
   int data;
   struct dll* prev;
   struct dll* next;
};
```

The key here is that each node points to TWO nodes, the one before it in the list AND the one after it.

The prev pointer in the first node is NULL while the next pointer in the last node is NULL.

Once again, much of the work here is similar to a regular linked list, except now we must manage and edit both forward and backward links. (Thus, there's more upkeep for each insertion or deletion.)

Let's take a look at inserting a node to the front of a DLL:

1) Create the new node and set both prev and next to NULL.

2) If the original list is empy, just return a pointer to this new node.

3) If not, set the next component of the new node to the original list.

4) Then, set the prev component of the front of the original list to point to the new node.

5) Return a pointer to the new node as the new pointer to the front of the list.

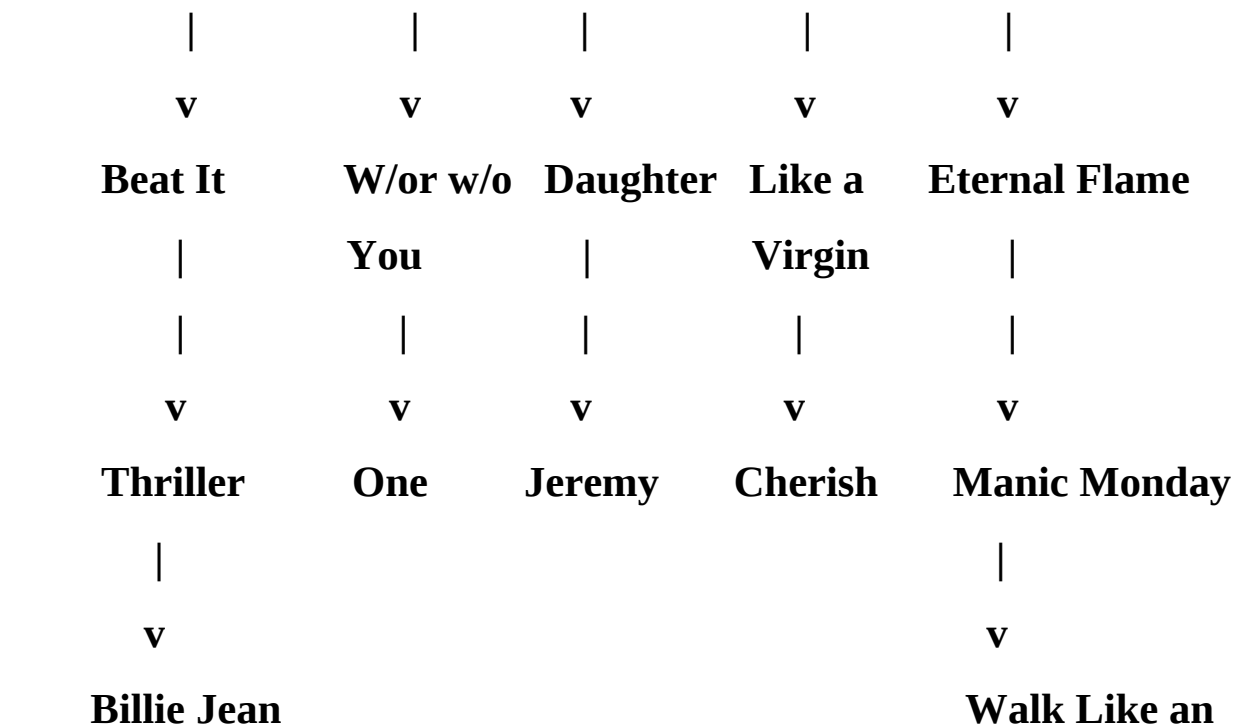
A linked list of linked lists: CD Example

Linked lists can be part of more complicated data structures.

Consider the situation where we have a linked list of musical artists:

→ Michael Jackson → U2 → Pearl Jam → Madonna → Bangles

It might make sense that each artist themselves has a linked list of songs (or albums) stored:



Egyptian

Here is how we can create the two necessary structs to maintain the structure dictated above:

```
struct CDNode {
   char title[50];
   struct CDNode* next;
};
```

```
struct ArtistNode {
      char first[30];
      char last[30];
      struct CDNode* listOfCDs;
      struct ArtistNode* next;
};
```