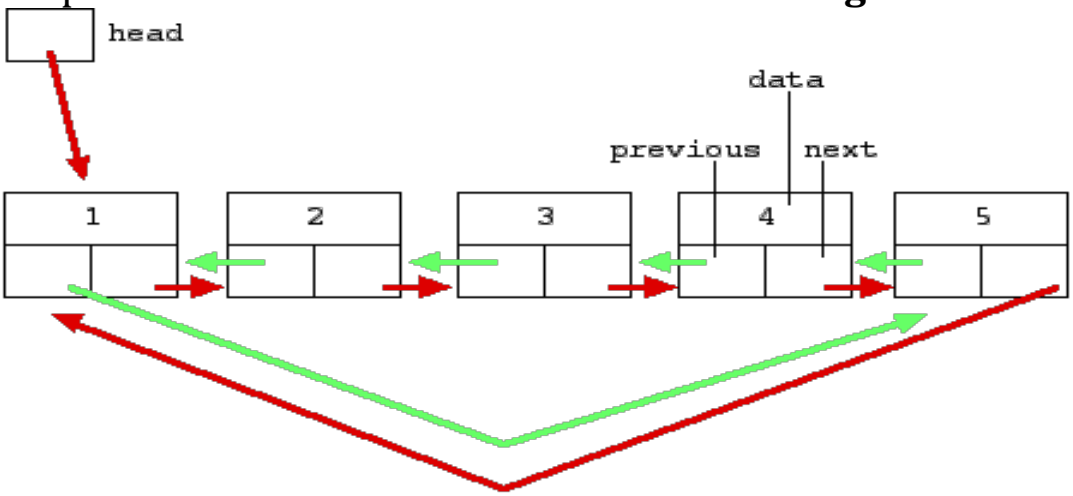
# CSS 2100 Project #2
# Songs Manager

## Introduction:

The program will manage a library of songs arranged by using a **doubly linked list**.

The doubly linked list is a sequential list of songs where each node is connected to the next one and the previous one. The first node is the head of the doubly linked list. The last node is not connected to the first one. The operations that can be done are **removeSong** and **addSong**.




**Doubly Linked List of songs**

## Tasks:

1) First, you will define the following structured data types:
    a. **Song**: contains the elements: int id, string name, string singerName.
    b. **SongNode**: contains the elements: Song sg, SongNode *previousNode, SongNode *nextNode.
    c. **SongDoublyLinkedList**: contains the element SongNode *firstElement.
2) Second, you will implement the following functions:
    a. **SongNode *getLastSong(SongDoublyLinkedList *songList):** the function will return the last song in songList if

it exists. The last node has the element NextNode equal to NULL.

b. **void addSong(SongDoublyLinkedList *songList, SongNode *songNd):** If songNd is the first element, the member firstElement of songList will take its value. Else, the function will link songNd to the last song of songList. A node n is linked to a node m of the list in the following way: n.previous will take the value of m and m.next will take the value of n.
c. **SongNode *removeSong(SongDoublyLinkedList *songList, int index):** If the node doesn't exist, the function will return NULL value. If the $node_{index}$ exists, it will be returned and the links of songList will be updated given the three following cases: $node_{index}$ doesn't have a next node, $node_{index}$ doesn't have previous node and $node_{index}$ has a previous node and a next one.
d. **void displayListElements(SongDoublyLinkedList *songList):** the function will display the Songs' names of songList starting from the first element of songList to the last element of songList.
e. **void clearList(SongDoublyLinkedList *songList):** the function will free the memory used by songList.

3) You will write the code of a **main.cpp** file where you will create a doubly linked list of songs. After that you will give the option to the user to use a function that you implemented and the choice to exist the program when he is done. After the execution of a function, the collection of data that was used has to be displayed to show the changes.