Before I implement my linked list, I noticed that we have given Node style:

```
typedef int Object;

struct LNode
{
    Object * data;
    LNode * next;
};
```

Since I am saving the object with address, I expect the user of my code must dynamically allocate the memory for the object, and send the address of the space as a parameter.

Furthermore, to convert my code to template easily, I use typedef operation.

```
class LinkedList
{
public:
    LinkedList();
    ~LinkedList();
    LinkedList(LinkedList& list);

    void insert(Object * data);
    bool append(const int index, Object * data);
    Object *  get(int i);
    Object * remove(int i);
    void reverseIterate();
    void reverseRecursive();
    Object * LFirst();
    Object * LNext();
    Object * LRemove();
    int getNumberOfData() const;

private:
    LNode * head;
    LNode * tail;
    LNode * cur;
    LNode * before;
    int numOfData;
};
```

The design of the program.

Since I am asked to implement a single linked list, I designed the linked list which does not have any dummy node.

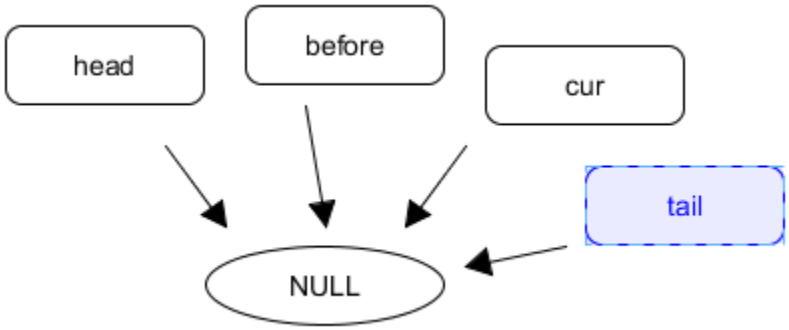
- Each node contains Object, and next node address.
- LinkedList class will contains head, tail, before, and cur node address.
- The number of data will be saved in the LinkedList.
- There would be constructor, and copy constructor.
- LFirst function will access the heading node.
- LNext Node will access the next node from the current node.
- Get(i) function will access the index I node. (get(0) will return the head node.)
- LRemove function will remove the current node.
- GetnumberofData function will return the number of data in the list.
- Reverseiterate and ReverseRecursive function will reverse the order of the list.

Note:

In order to use the code efficiently, a programmer who invoke my code must know that for the optimal performance of my code, I encourage the programmer to use LFirst, and LNext function instead of get(i) function.

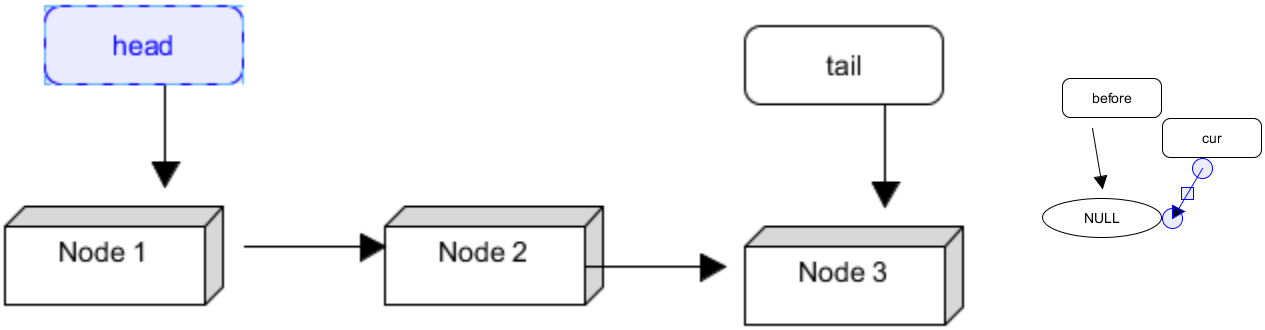
With constructor, I make head, cur, before, and number of the data equal to 0(NULL). When I add data, I design that tail add data. In UML Diagram.

Constructor called:

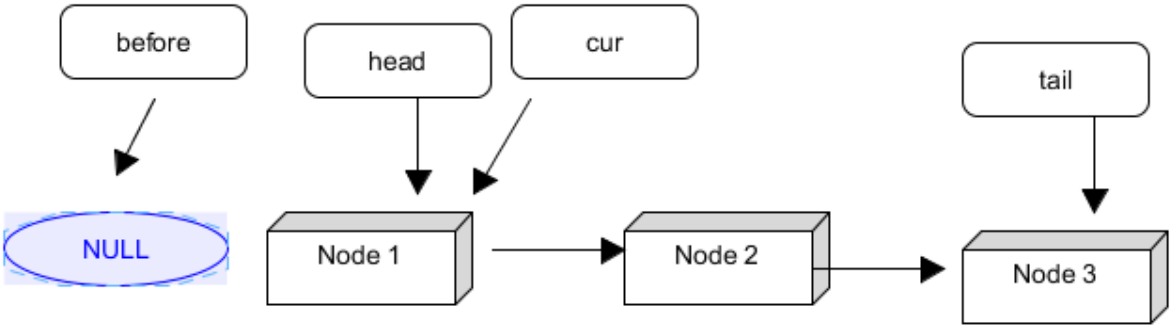


and num of data = 0.

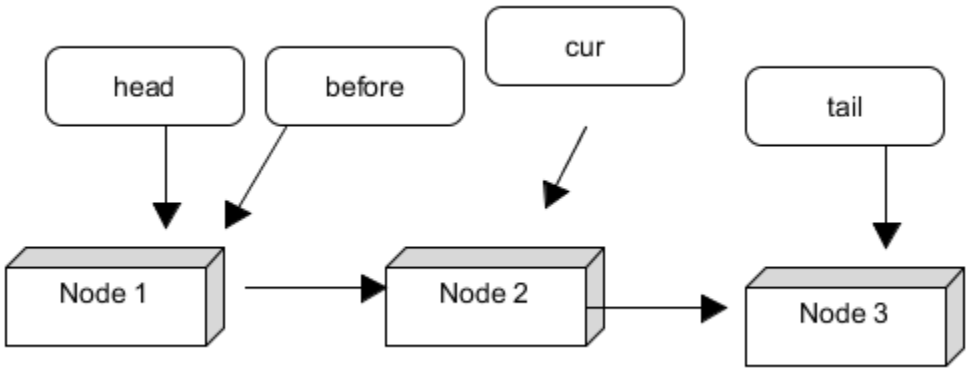
When the user insert data, I assume that the user input 3rd object element into the list then



When the LFirst is invoked then (Node 3 next is null)



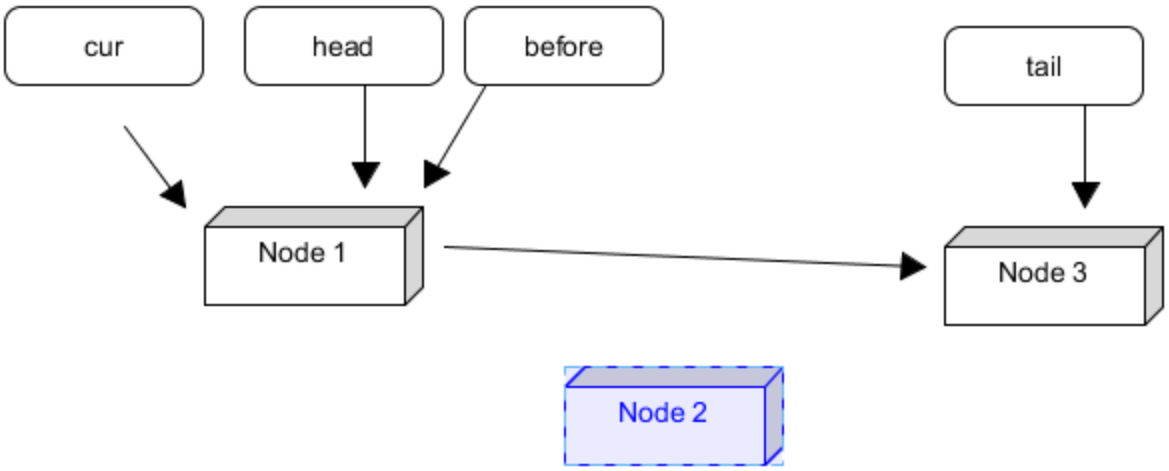
When LNext is invoked (node 3 next is null)



The Designed the recursive and iterate reverse function differently, and description is visually illustrated.

Note: Since there are three node, the Node three next is indicating NULL!

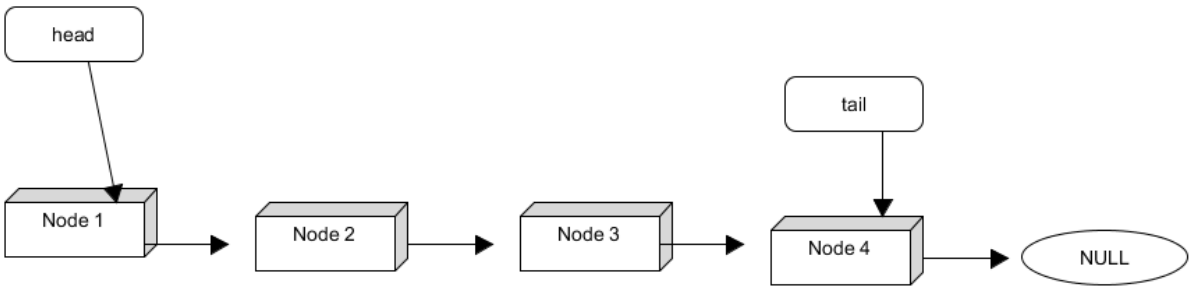
When we invoke the LRemove function when the current pointer is indicating the Node_2, then



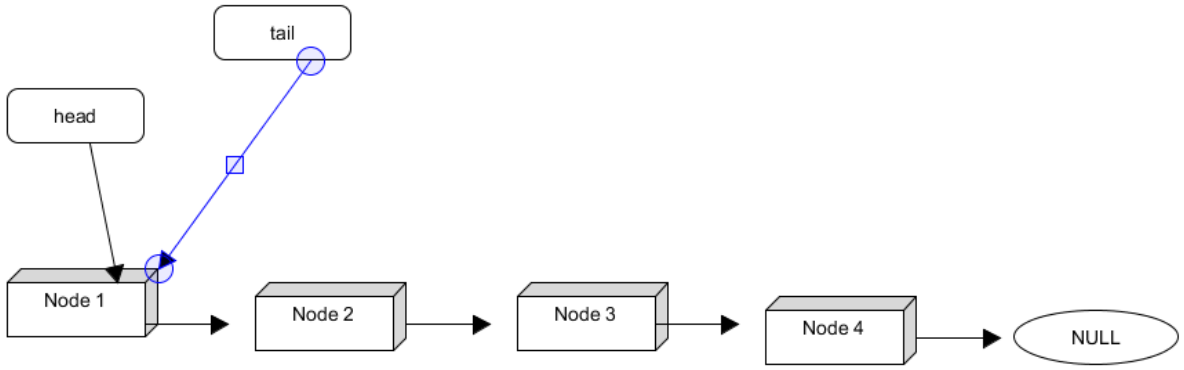
Node2's data(object) should be deleted by the programmer who invokes my code, otherwise, the Object, the programmer dynamically allocated, would not be deleted.

Reversing the list. (Two approaches)

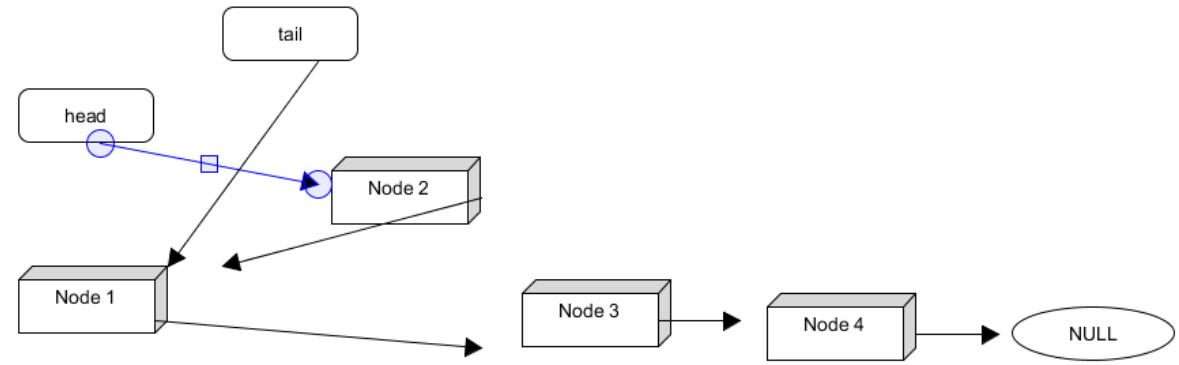
Iterate Version of reverse

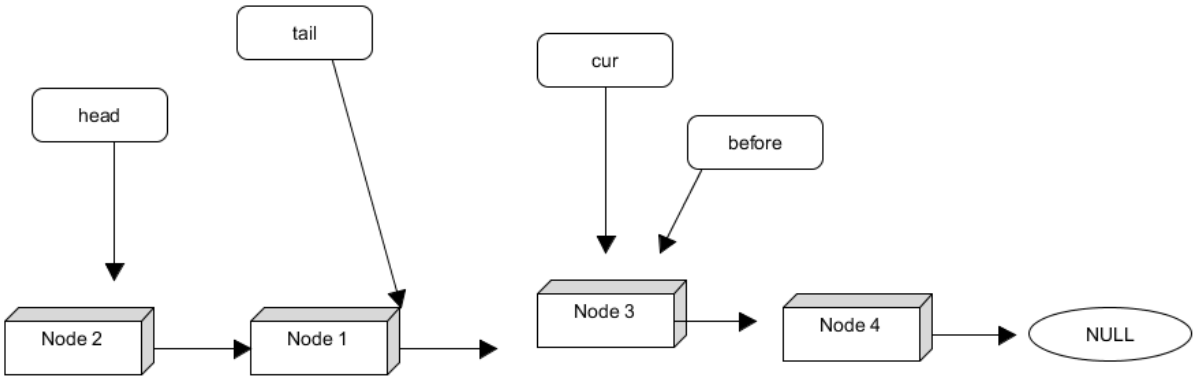


In order to reverse it iteratively, I set the tail node pointer to point the heading node, and move the new tail node to the tail by rearranging the node.



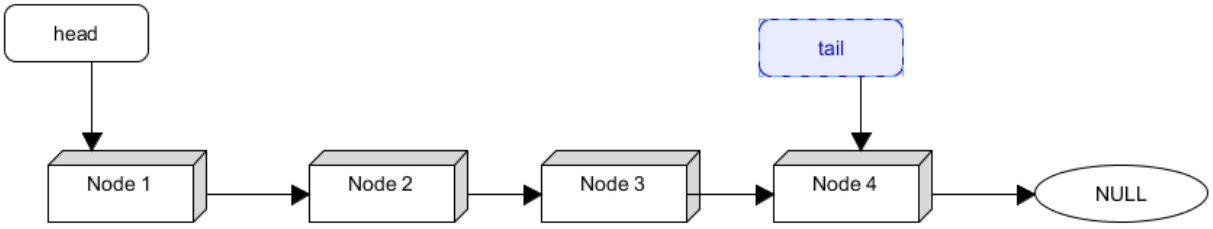
Node operation at the first loop





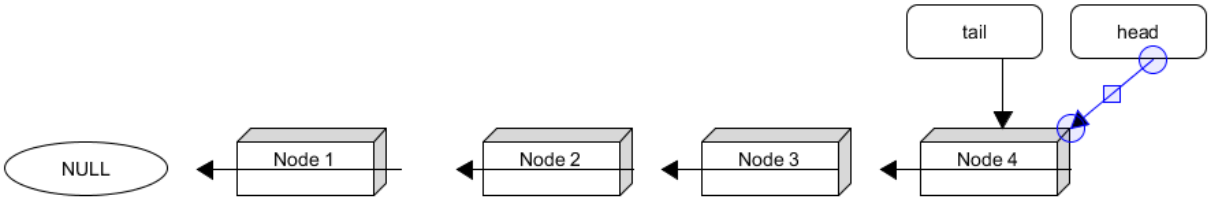
Do this process iteratively until the $tail \rightarrow next = null$

For the recursive function

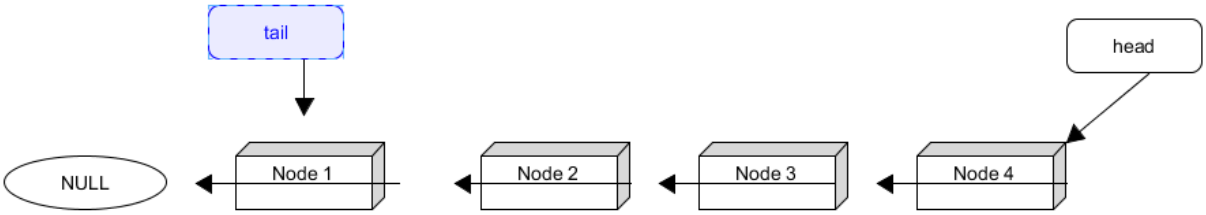


It starts from here.

At first, I move the head pointer to point the node 4 while changing the direction of each node is reversed.



I move the tail node to the node 1.



This is reversing operation of the recursive version.

Discussion about the differences and similarities between the two approaches, i.e., iterative vs. Recursive.

Similarities.

- Repeat the instruction over and over, till the break condition(s) meet(s).
- Must have break statement(escape condition) otherwise, the program will never exit.

Differences

- Recursive function takes more memory compared to integrate statement.
- The value of the variable should be passed as a parameter when we use the recursive function, or must be defined in the function, while the iterate loop keeps the variable. (However, if I make a global variable, the recursive function can be invoked as a iterate loop.)