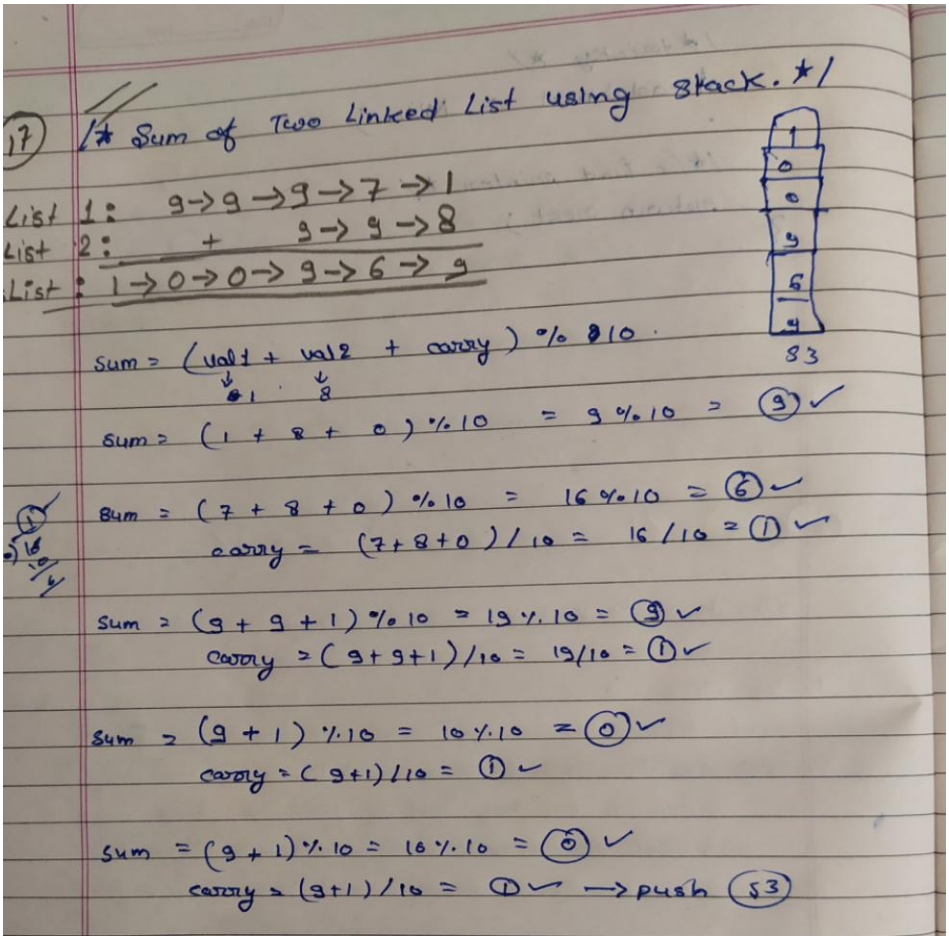
Sum of two linked list using Stack



Given two numbers which are represented using linked lists as shown below. Your program should return the reference to a new linked list which stores the sum of given two numbers.

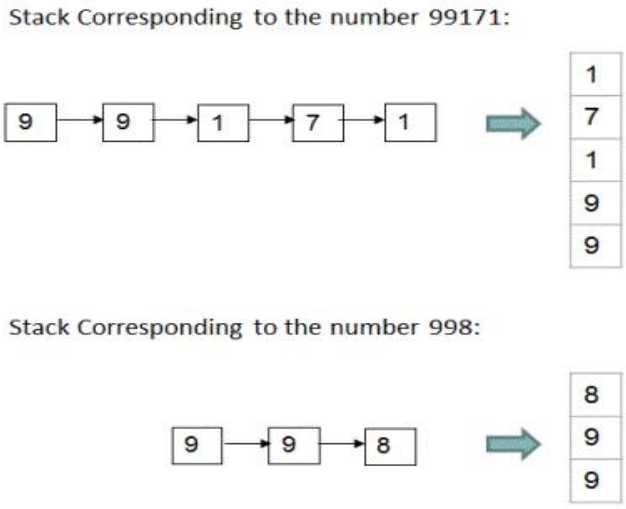
Numbers are represented as following:
Number 99971, corresponding linked list: 9->9->9->7->1
Number 998, corresponding linked list: 9->9->8

The output returned by the program for above two linked lists as the input should be the linked list 1->0->0->9->6->9 which represents number 100969 which is sum of numbers 99971 and 998.

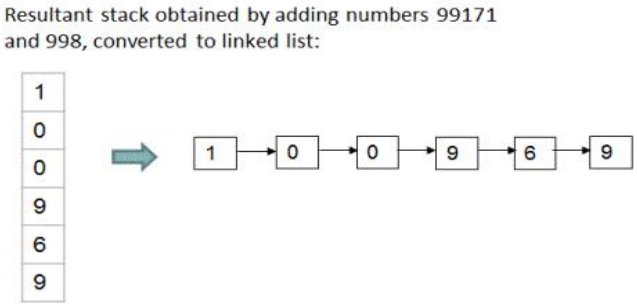
Algorithm/Insights

Here we will be discussing a non-recursive algorithm to add two numbers represented using linked lists.

Idea: Because the unit digits of numbers are placed at the end of the linked lists which represent them, to add given two numbers, we need to start adding the values of the nodes starting from the end nodes of linked lists. For example, to add numbers 99,171(9->9->9->7->1) and 998(9->9->8), we start by adding node with value '1' of the first list to the node with value '8' of the second list. To enforce this reverse ordering, we can push the given two lists to two different stacks thereby placing the unit digits at the top of the stacks and then starting the addition process by popping digits from the stacks one by one. Also since we need to return the resultant number represented in the form of a linked list(which has unit digit at the end), we push the addition result onto a stack and create the linked list out of this stack. For adding two numbers 99,171 and 998, the stacks will look like following after pushing the two linked lists to stacks.



And the result of addition of these two numbers would again be pushed to stack digit by digit. This resultant stack then would be converted to the linked list.



Algorithm: The formal steps of this algorithm are as following:
1. Create stack 's1' by pushing all node values of the first linked list to a stack.
2. Create stack 's2' by pushing all node values of the second linked list to a stack.
3. Create an empty stack 's3' to store the result of addition.
4. Initialize sum and carry to 0.
5. Pop the top element from stack 's1'. Let this top element be 'value1'.
6. Pop the top element from stack 's2'. Let this top element be 'value2'.
7. Make 'sum' = (value1 + value2 + carry)%10 and push this 'sum' to stack 's3' and make 'carry' = (value1 + value2 + carry)/10.
8. Repeat steps 5-7 till one of the stacks become empty. If both stacks are of same size, then both of the stacks would become empty at the same time.
9. If stack 's1' has elements left in it then -
    a. Pop the top element from stack 's1'. Let this top element be 'value1'.
    b. Make 'sum' = (value1 + carry)%10 and push this 'sum' to stack 's3' and make 'carry' = (value1 + carry)/10.
    c. Repeat steps 9a and 9b until stack 's1' is not empty.
10. Similarly, if stack 's2' has elements left in it then -
    a. Pop the top element from stack 's2'. Let this top element be 'value2'.
    b. Make 'sum' = (value2 + carry)%10 and push this 'sum' to stack 's3' and make 'carry' = (value2 + carry)/10.
    c. Repeat steps 10a and 10b until stack 's2' is not empty.
11. After all the above steps are executed, if carry is greater than 0, push it to the resultant stack 's3'.
12. Create an empty linked list 'result'. Now pop elements one by one from the stack 's3', and keep appending them to the 'result' linked list until stack 's3' is not empty. Return the output as 'result' linked list.