Merit-Quality-Excellence

Sukkur IBA University Khairpur Campus

Data Structures

LAB No: 05

Instructor: Marina Gul

Objective of Lab No. 5:
After performing lab 5, students will be able to:

- More on Linked List
- More on Stack
- More on Queue

Practice 01: Reverse a Linked List

Given a linked list, the task is to reverse the linked list by changing the links between nodes.

Examples:

Input: Linked List = 1 -> 2 -> 3 -> 4 -> NULL
Output: Reversed Linked List = 4 -> 3 -> 2 -> 1 -> NULL

Input: Linked List = 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Output: Reversed Linked List = 5 -> 4 -> 3 -> 2 -> 1 -> NULL

Input: Linked List = NULL
Output: Reversed Linked List = NULL

Input: Linked List = 1->NULL
Output: Reversed Linked List = 1->NULL

Using Iterative Method – O(n) Time and O(1) Space:

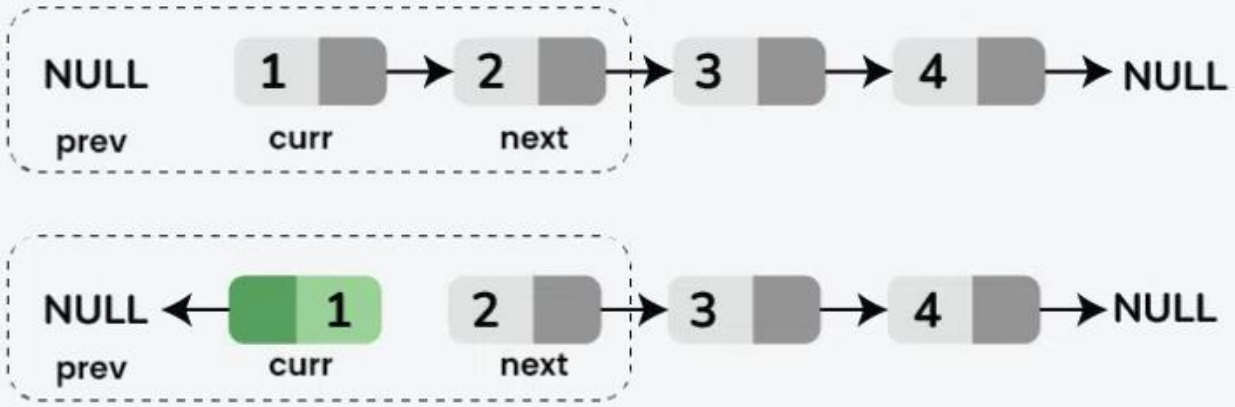
The idea is to reverse the links of all nodes using three pointers:

- prev: pointer to keep track of the previous node
- curr: pointer to keep track of the current node
- next: pointer to keep track of the next node

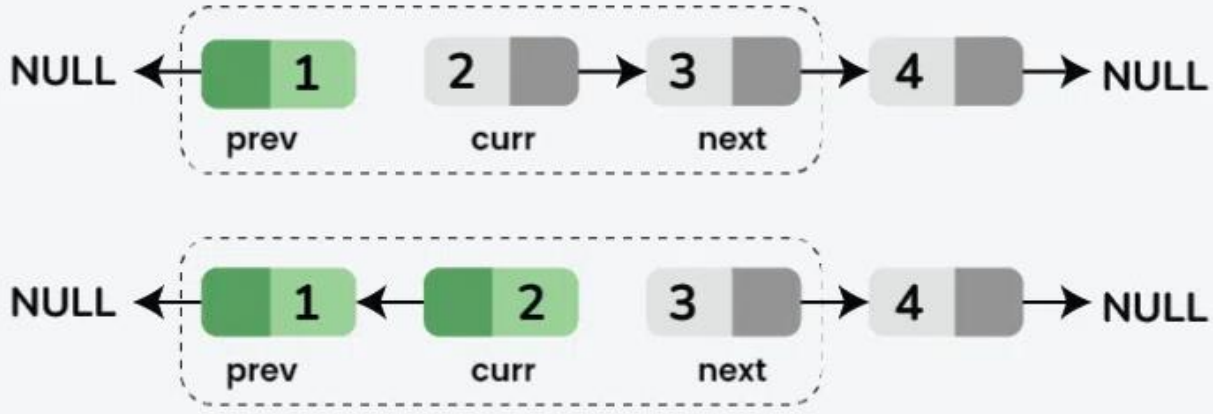
Starting from the first node, initialize curr with the head of linked list and next with the next node of curr. Update the next pointer of curr with prev. Finally, move the three pointer by updating prev with curr and curr with next.

01 Step | Input Linked List having 4 nodes

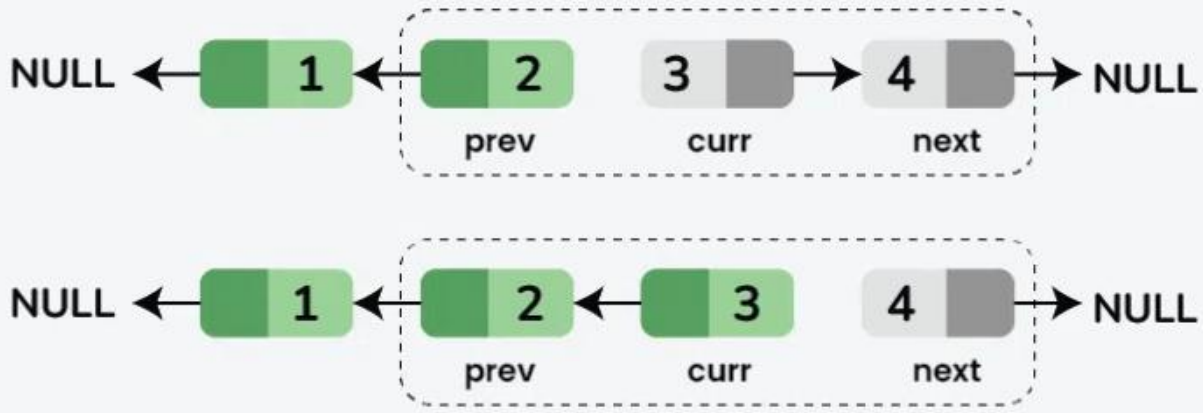
02 Step | Initialize prev pointer = NULL
Store next = curr.next
Update curr.next = prev



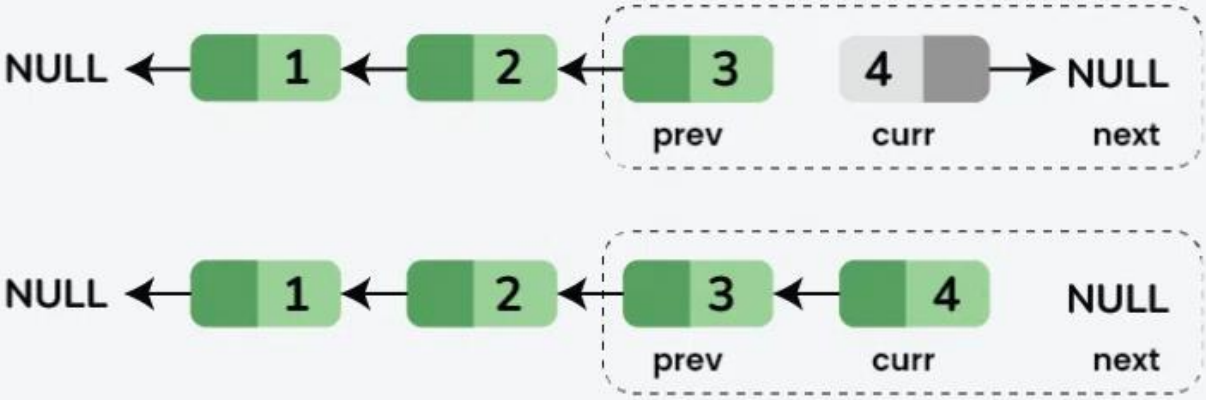
03 Step | Update prev = curr and curr = next
Store next node 3 as next
Update next pointer of current node 2 to previous node 1.

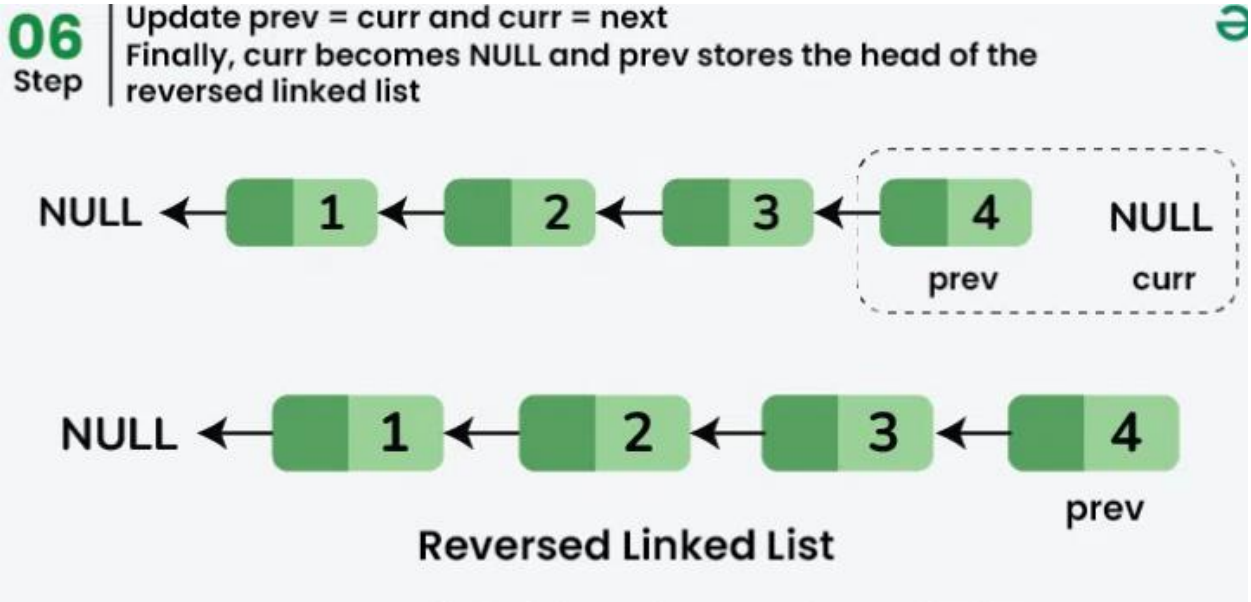


04 Step | Update prev = curr and curr = next
Store next node 4 as next
Update next pointer of current node 3 to previous node 2.



05 Step | Update prev = curr and curr = next
next pointer points to NULL
Update next pointer of current node 4 to previous node 3.

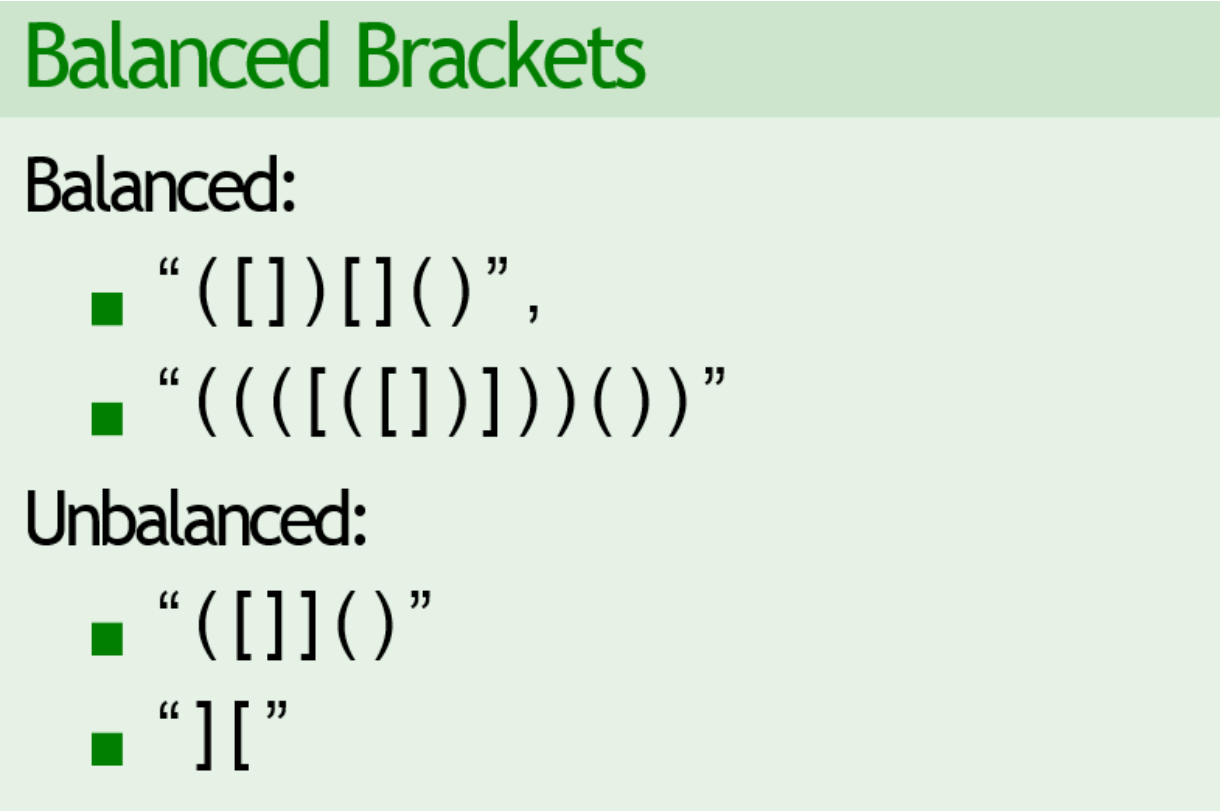




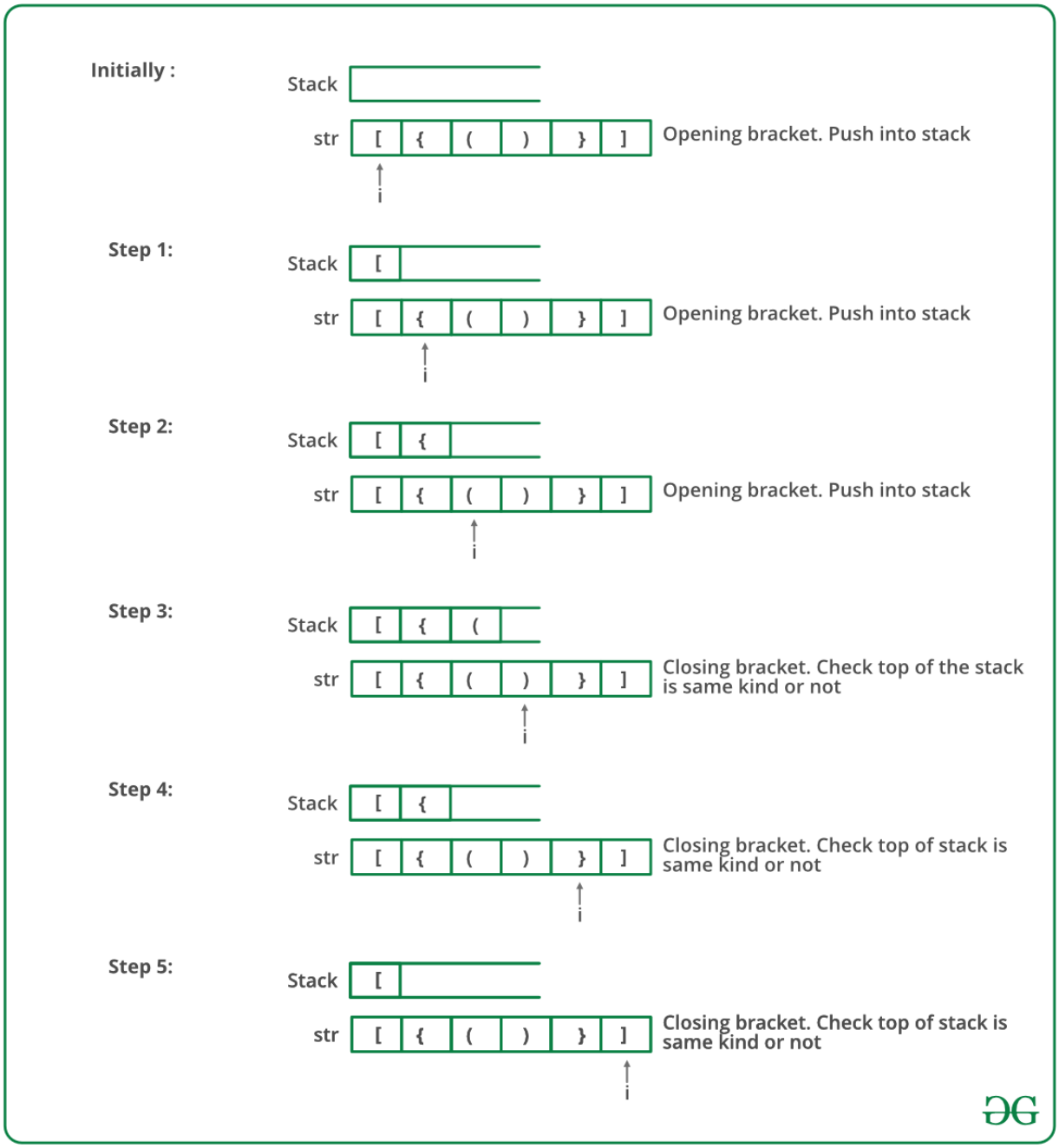
Read more

Balanced Brackets Problem

Given an expression string exp, write a program to examine whether the pairs and the orders of “{“, “}”, “(“, “)”, “[“, “]” are correct in the given expression.



The idea is to put all the opening brackets in the stack. Whenever you hit a closing bracket, search if the top of the stack is the opening bracket of the same nature. If this holds then pop the stack and continue the iteration. In the end if the stack is empty, it means all brackets are balanced or well-formed. Otherwise, they are not balanced.



Algorithm:

1. Declare a character stack (say temp).
2. Now traverse the string exp.
3. If the current character is a starting bracket (‘(‘ or ‘{‘ or ‘[‘) then push it to stack.
4. If the current character is a closing bracket (‘)’ or ‘}’ or ‘]’) then pop from the stack and if the popped character is the matching starting bracket then fine.
5. Else brackets are Not Balanced.
6. After complete traversal, if some starting brackets are left in the stack then the expression is Not balanced, else Balanced.

Infix expression to Postfix expression

The compiler scans the expression either from left to right or from right to left.

Consider the expression: **a + b * c + d**

The compiler first scans the expression to evaluate the expression b * c, then again scans the expression to add a to it. The result is then added to d after another scan.

The repeated scanning makes it very inefficient. Infix expressions are easily readable and solvable by humans whereas the computer cannot differentiate the operators and parenthesis easily so, it is better to convert the expression to postfix(or prefix) form before evaluation.

The corresponding expression in postfix form is abc*+d+. The postfix expressions can be evaluated easily using a stack.

How to convert an Infix expression to a Postfix expression?

To convert infix expression to postfix expression, use the stack data structure. Scan the infix expression from left to right. Whenever we get an operand, add it to the postfix expression and if we get an operator or parenthesis add it to the stack by maintaining their precedence.

More read

Exercise

1. Print in reverse order: You are asked to design a method in linked list to print data in reverse order. You don't need to reverse linked list permanently.

```
Public void printReverse()
```

2. Balanced Brackets: Take user string input and check whether it's balanced or not. Use stack functions. Input may contain any of the bracket among {, [, (and any number and letters like: ({[a+b]+c}-1) and so on.

3. FirstSingleLetter: Create the function **char firstSingleLetter (const char text [], const int n)** which finds and returns the first letter of text that occurs only once. n is the number of characters in the text.

Here in this task, it is not allowed to use any help structures - like for example. linkedlist, stack, queue or list. The function will therefore be as O (n^2). Text contains only the letters A-Z (only uppercase letters, and no spaces).

Input: "algorithm"
Output: a
Explanation: As 'a' is first character in the string which does not repeat.

4. Convert **Infix expression to Postfix expression** using Stack data structure.

Input: *A + B * C + D*
Output: *ABC*+D+*