DAY 23

INTERVIEW BIT PROBLEMS :

1. Generate all Parentheses

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.
The brackets must close in the correct order, "()" and "()[]{}" are all valid
but "(]" and "([)]" are not.
Return 0 / 1 (0 for false, 1 for true) for this problem

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @return an integer
    def isValid(self, A):
        paran_dict={'{':'}','(':')','[':']'}
        if len(A)==0:
            return 1
        if len(A)==1 or(A[0] not in paran_dict):
            return 0
        stack=[]
        for c in A:
            if c in paran_dict:
                stack.append(c)
            elif len(stack)==0:
                return 0
            else:
                if c==paran_dict[stack[-1]]:
                    stack.pop()
                else:
                    return 0
        if len(stack)==0:
            return 1
        else:
            return 0
```

JAVASCRIPT

```
module.exports = {
 //param A : string
 //return an integer
    isValid : function(A){
        params_dict={'{':'}','(':')','[':']'};
        stack=[];
        for(var c=0;c<A.length;c++){
```

```
            if(A[c]=='{' || A[c]=='[' || A[c]=='('){
                stack.push(A[c]);
            }
            else if(A[c]==')' || A[c]=='}' || A[c]==']'){
                var check=stack.pop();
                if(params_dict[check]!=A[c]){
                    return 0;
                }
            }
        }
        return stack.length>0?0:1;
    }
};
```

2. Swap List Nodes in pairs

Given a linked list, swap every two adjacent nodes and return its head.
For example,
Given 1->2->3->4, you should return the list as 2->1->4->3.
Your algorithm should use only constant space. You may not modify the values in the list, only nodes itself can be changed.

CODE :

PYTHON

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    # @param A : head node of linked list
    # @return the head node in the linked list
    def swapPairs(self, A):
        temp=A
        if temp is None:
            return None
        while temp is not None and temp.next is not None:
            temp.val,temp.next.val=temp.next.val,temp.val
            temp=temp.next.next
        return A
```

JAVASCRIPT :

```
// Definition for singly-linked list.
//     function Node(data){
//         this.data = data
//         this.next = null
```

```
//     }

module.exports = {
 //param A : head node of linked list
 //return the head node in the linked list
    swapPairs : function(A){
        if (A == null){
            return null;
        }
        let temp = A;
        while (temp != null && temp.next != null) {
            let t = temp.data;
            temp.data = temp.next.data;
            temp.next.data = t;
            temp = temp.next.next;
        }
        return A;
    }
};
```

C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *      int val;
 *      ListNode *next;
 *      ListNode(int x) : val(x), next(NULL) {}
 * };
 */
ListNode* Solution::swapPairs(ListNode* A) {
        ListNode* temp=A;
        while(temp!=NULL && temp->next!=NULL){
            swap(temp->val,temp->next->val);
            temp=temp->next->next;
        }
        return A;
}
```