



---

# ADVANCED CODING 2

---

## ASSIGNMENT 2



VU21CSENo300277  
PHANINDRA RAVIPATI

## 1. Maximum Sum Circular Subarray

```
class Solution {  
    public int maxSubarraySumCircular(int[] nums) {  
        int n = nums.length;  
        int maxSum = nums[0], minSum = nums[0], curMax=0, curMin=0;  
        int totalSum = 0;  
        for(int num : nums){  
            curMax = Math.max(curMax + num, num);  
            maxSum = Math.max(maxSum, curMax);  
            curMin = Math.min(curMin + num, num);  
            minSum = Math.min(minSum, curMin);  
            totalSum+=num;  
        }  
        return maxSum>0? Math.max(maxSum, totalSum-minSum) : maxSum;  
    }  
}
```

OUTPUT:

☑ Testcase | >\_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

```
nums =  
[1,-2,3,-2]
```

Output

3

Expected

3

☑ Testcase | >\_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

```
nums =  
[5,-3,5]
```

Output

10

Expected

10

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =  
[-3,-2,-3]

Output

-2

Expected

-2

## 2. Stamping The Sequence

```
class Solution {
    public int[] movesToStamp(String S, String T) {
        if (S == T) return new int[]{0};
        char[] SC = S.toCharArray(), TC = T.toCharArray();
        int slen = SC.length, tlen = TC.length - slen + 1, i, j;
        List<Integer> lans = new ArrayList<>();
        Boolean tdiff = true, sdiff;
        while (tdiff)
            for (i = 0, tdiff = false; i < tlen; i++) {
                for (j = 0, sdiff = false; j < slen; j++)
                    if (TC[i+j] == '*') continue;
                    else if (TC[i+j] != SC[j]) break;
                    else sdiff = true;
                if (j == slen && sdiff) {
                    for (j = i, tdiff = true; j < slen + i; j++)
                        TC[j] = '*';
                    lans.add(i);
                }
            }
        for (i = 0; i < TC.length; i++) if (TC[i] != '*') return new int[]{};
        int[] ans = new int[lans.size()];
        for (i = 0; i < lans.size(); i++) ans[i] = lans.get(i);
        return ans;
    }
}
```

## OUTPUT

• Case 1

• Case 2

Input

stamp =  
"abca"

target =  
"aabcaca"

Output

[0,3,1]

Expected

[3,0,1]

• Case 1

• Case 2

Input

stamp =  
"abc"

target =  
"ababc"

Output

[0,2]

Expected

[0,2]

### 3. Design Browser History

```
class BrowserHistory {
```

```
    public class Node{  
        String url;  
        Node next, prev;  
        public Node(String url) {  
            this.url = url;  
            next = null;  
            prev = null;  
        }  
    }
```

```
    Node curr;  
    public BrowserHistory(String homepage) {
```

```

    curr = new Node(homepage);
}

public void visit(String url) {
    Node node = new Node(url);
    curr.next = node;
    node.prev = curr;
    curr = node;
}

public String back(int steps) {
    while (curr.prev != null && steps-- > 0) {
        curr = curr.prev;
    }
    return curr.url;
}

public String forward(int steps) {
    while (curr.next != null && steps-- > 0) {
        curr = curr.next;
    }
    return curr.url;
}
}

```

## OUTPUT

Case 1

Input

["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]

["leetcode.com"],["google.com"],["facebook.com"],["youtube.com"],[1],[1],[1],["linkedin.com"],[2],[2],[7]]

Output

[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","leetcode.com"]

Expected

[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","leetcode.com"]

## 4. LRU Cache

```
import java.util.HashMap;

class LRUCache {
    class Node {
        int key, value;
        Node prev, next;

        Node(int key, int value) {
            this.key = key;
            this.value = value;
        }
    }

    private final int capacity;
    private final HashMap<Integer, Node> map;
    private final Node head, tail;

    public LRUCache(int capacity) {
        this.capacity = capacity;
        map = new HashMap<>();
        head = new Node(-1, -1);
        tail = new Node(-1, -1);
        head.next = tail;
        tail.prev = head;
    }

    private void addNode(Node newNode) {
        newNode.next = head.next;
        newNode.prev = head;
        head.next.prev = newNode;
        head.next = newNode;
    }

    private void removeNode(Node node) {
        node.prev.next = node.next;
        node.next.prev = node.prev;
    }

    public int get(int key) {
        if (map.containsKey(key)) {
            Node node = map.get(key);
            removeNode(node);
            addNode(node);
            return node.value;
        }
        return -1;
    }
}
```

```

public void put(int key, int value) {
    if (map.containsKey(key)) {
        Node node = map.get(key);
        removeNode(node);
        map.remove(key);
    }
    if (map.size() == capacity) {
        Node lruNode = tail.prev;
        removeNode(lruNode);
        map.remove(lruNode.key);
    }
    Node newNode = new Node(key, value);
    addNode(newNode);
    map.put(key, newNode);
}
}

```

## OUTPUT

Accepted
Runtime: 0 ms

• Case 1

Input

["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]

[[2], [1,1], [2,2], [1], [3,3], [2], [4,4], [1], [3], [4]]

Output

[null,null,null,1,null,-1,null,-1,3,4]

Expected

[null,null,null,1,null,-1,null,-1,3,4]