DAY6-19/11/2024

1.Longest Substring without Repeating characters

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        int maxLen = 0;
        int l = 0, r = 0;
        unordered_set<char> chars;
        while(r<s.size()){
            while(chars.find(s[r]) != chars.end()){
                chars.erase(s[l]);
                ++l;
            }
            maxLen = max(maxLen,r-l+1);
            chars.insert(s[r]);
            ++r;
        }
        return maxLen;
    }
};
```

2.Spiral Matrix

```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        int m = matrix.size();
        vector<int> res;
        int n = matrix[0].size();
        int top = 0, bottom = m - 1, left = 0, right = n - 1;
        while (top <= bottom && left <= right) {
            for (int i = left; i <= right; i++) res.push_back(matrix[top][i]);
            top++;
            for (int i = top; i <= bottom; i++) res.push_back(matrix[i][right]);
            right--;
            if (top <= bottom) {
                for (int i = right; i >= left; i--) res.push_back(matrix[bottom][i]);
                bottom--;
            }
            if (left <= right) {
                for (int i = bottom; i >= top; i--) res.push_back(matrix[i][left]);
                left++;
            }
        }
        return res;
    }
};
```

3.Remove linked list elements

```
class Solution {
public:
    ListNode* removeElements(ListNode* head, int val) {
        while(head != NULL && head->val == val) {
            head = head->next;

        }
        if(head == NULL){
            return NULL;
        }
        ListNode* curr = head;

        while(curr != NULL && curr->next != NULL ){
            if(curr->next->val == val){
                curr->next = curr->next->next;
            }
            else{
                curr = curr->next;
            }
        }
        return head;

    }
};
```

4.linked list palindrome

```
class Solution {
public:
    bool isPalindrome(ListNode* head) {
        stack<int> st;
        if(head == nullptr)
        {
            return false;
        }
        ListNode* curr = head;
        while(curr != nullptr)
        {
            st.push(curr->val);
            curr = curr->next;
        }
        curr = head;
        while(curr != nullptr)
        {
            if(st.top() != curr->val)
            {
                return false;
            }
            st.pop();
            curr = curr->next;
        }
        return true;

    }
};
```

5.Next permutation

```
class Solution {
public:
    void nextPermutation(vector<int>& nums) {
         next_permutation(nums.begin(), nums.end());
    }
};
```