```python
class Solution:
    def isValid(self, s: str) -> bool:
        stack=[]

        for i in s:

            if i not in ")}]":
                stack.append(i)

            elif stack:
                x=stack.pop()

                if((x=="(" and i!=")") or (x=="{" and i!="}") or (x=="[" and i!="]")):
                    return False

            else:
                return False

        return False if stack else True
```

```python
class Solution(object):
    def nextGreaterElement(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: List[int]
        """
        ans=[]

        for i in nums1:
            a=nums2.index(i)
            if a+1 != len(nums2) :
                count=0
                for j in nums2[a+1: ] :
                    if j>i :
                        ans=ans+[j]
                        count=1
                        break

                if count==0:
                    ans=ans+[-1]

            else:
                ans=ans+[-1]

        return ans
```

i C++ Auto

```cpp
class Solution {
public:
    string removeDuplicateLetters(string s) {
        vector<int> freq(26,0);
        for(int i=0 ; i<s.size() ; ++i)
        freq[s[i]-'a']++;

        stack<char> st;
        vector<bool> seen(26 , false);
        for (int i=0 ; i< s.size() ; ++i){
            if (seen[s[i]-'a']){
                freq[s[i]-'a']--;
                continue;
            }
            while(!st.empty() and st.top()>s[i] and freq[st.top()-'a']>0){
                seen[st.top()-'a']=false;
                st.pop();
            }
            st.push(s[i]);
            seen[s[i]-'a']=true;
            freq[s[i]-'a']--;
        }
        string ans="";
        while (!st.empty()){
            ans.push_back(st.top());
            st.pop();
        }
        reverse(ans.begin(), ans.end());
        return ans;

    }
};
```

```cpp
class MyQueue {
public:
    stack <int> s1;
    stack <int> s2;
    MyQueue() {

    }

    void push(int x) {
        while (!s1.empty()){
            s2.push(s1.top());
            s1.pop();
        }
        s2.push(x);
        while (!s2.empty()){
            s1.push(s2.top());
            s2.pop();
        }

    }

    int pop() {
        int curr = s1.top();
        s1.pop();
        return curr;
    }

    int peek() {
        return s1.top();
    }

    bool empty() {
        return s1.empty();
    }
};
```

```cpp
class DataStream {
    int val;
    int n ;
    int count;
public:
    DataStream(int value, int k) {
        val=value;
        n=k;
        count=0;

    }

    bool consec(int num) {
        if (num==val){
            count++;
        }
        else{
            count=0;
            return false;
        }
        if (count==n){
            count=n-1;
            return true;
        }
        return false;
    }
};
```