```cpp
class Solution {
public:
    int maxArea(vector<int>& height) {
        int left = 0;
        int right = height.size() - 1;
        int maxArea = 0;

        while (left < right) {
            int currentArea = min(height[left], height[right]) * (right - left);
            maxArea = max(maxArea, currentArea);

            if (height[left] < height[right]) {
                left++;
            } else {
                right--;
            }
        }

        return maxArea;
    }
};
```


Testcase | Test Result

Case 1 Case 2 +

height =

```
[1,8,6,2,5,4,8,3,7]
```

```cpp
//Approach 1 :
//time complexity - O(1) since the algorithm always iterates through a constant number of values (13 in this case).
//O(1) since the amount of extra space used is constant (the size of the storeIntRoman vector, which is also 13 in this case
class Solution {
public:
    string intToRoman(int num) {
        string Roman = "";
        // Creating vector of pairs to store the Roman numeral values and their corresponding symbols
        vector<pair<int, string>> storeIntRoman = {{1000, "M"}, {900, "CM"}, {500, "D"}, {400, "CD"}, {100, "C"}, {90, "XC"}, {50, "L"}, {40, "XL"}, {10, "X"}, {9, "IX"}, {5, "V"}, {4, "IV"}, {1, "I"}};
        // Iterating through the vector and repeatedly adds the corresponding symbols to the result string while subtracting the corresponding value from the input integer until the input integer becomes zero.
        for (int i = 0; i < storeIntRoman.size(); i++) {
            while (num >= storeIntRoman[i].first) {
                Roman += storeIntRoman[i].second;
                num -= storeIntRoman[i].first;
            }
        }
        return Roman;
    }
};
```

Saved

Ln 19, C

Testcase | Test Result

Accepted Runtime: 2 ms

• Case 1 • Case 2 • Case 3

Input

num =

3749

Output

"MMMDCCXLIX"

```cpp
class Solution {
public:
    int romanToInt(string s) {
        unordered_map<char, int> m;

        m['I'] = 1;
        m['V'] = 5;
        m['X'] = 10;
        m['L'] = 50;
        m['C'] = 100;
        m['D'] = 500;
        m['M'] = 1000;

        int ans = 0;

        for(int i = 0; i < s.length(); i++){
            if(m[s[i]] < m[s[i+1]]){
                ans -= m[s[i]];
            }
            else{
                ans += m[s[i]];
            }
        }
        return ans;
    }
};
```

```
        m['I'] = 1;
        m['V'] = 5;
        m['X'] = 10;
        m['L'] = 50;
        m['C'] = 100;
        m['D'] = 500;
        m['M'] = 1000;

        int ans = 0;

        for(int i = 0; i < s.length(); i++){
            if(m[s[i]] < m[s[i+1]]){
                ans -= m[s[i]];
            }
            else{
                ans += m[s[i]];
            }
        }
        return ans;
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 4 ms

• Case 1 • Case 2 • Case 3

Input

s =

"III"

Output

3

```cpp
class Solution {
public:
    string longestCommonPrefix(vector<string>& v) {
        string ans="";
        sort(v.begin(),v.end());
        int n=v.size();
        string first=v[0],last=v[n-1];
        for(int i=0;i<min(first.size(),last.size());i++){
            if(first[i]!=last[i]){
                return ans;
            }
            ans+=first[i];
        }
        return ans;
    }
};
```

Saved

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

strs =

```
["flower","flow","flight"]
```

Output

```
"fl"
```

```cpp
//Optimized Approach - O(n^2 logn + nlogn) - o(n^2 logn) time and O(n) space
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        int target = 0;
        sort(nums.begin(), nums.end());
        set<vector<int>> s;
        vector<vector<int>> output;
        for (int i = 0; i < nums.size(); i++){
            int j = i + 1;
            int k = nums.size() - 1;
            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == target) {
                    s.insert({nums[i], nums[j], nums[k]});
                    j++;
                    k--;
                } else if (sum < target) {
                    j++;
                } else {
                    k--;
                }
            }
        }
        for(auto triplets : s)
            output.push_back(triplets);
        return output;
    }
};
```

```cpp
        sort(nums.begin(), nums.end());
        set<vector<int>> s;
        vector<vector<int>> output;
        for (int i = 0; i < nums.size(); i++){
            int j = i + 1;
            int k = nums.size() - 1;
            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == target) {
                    s.insert({nums[i], nums[j], nums[k]});
                    j++;
                    k--;
                } else if (sum < target) {
                    j++;
                } else {
                    k--;
                }
            }
        }
        for(auto triplets : s)
            output.push_back(triplets);
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =

[-1,0,1,2,-1,-4]

Output

[[-1,-1,2],[-1,0,1]]

```cpp
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        int len = nums.size();
        if(len == 3) return nums[0] + nums[1] + nums[2];

        sort(nums.begin(), nums.end());

        int ans = nums[0] + nums[1] + nums[2];
        if(ans >= target) return ans;

        int max = nums[len - 1] + nums[len - 2] + nums[len - 3];
        if(max <= target) return max;

        int last = nums[0];
        int left, right, sum, num;
        int dist = abs(ans - target);
        int i = 0;

        for(; i < len - 2; i++){
            if (i && nums[i] == last)
                continue;
            last = num = nums[i];
            left = i + 1;
            right = len - 1;
            while(left < right){
                sum = num + nums[left] + nums[right];
                if (sum == target) return sum;
                if(abs(sum - target) < dist){
                    ans = sum;
                    dist = abs(ans - target);
                }
                if(sum < target){
                    while(left < right && nums[left] == nums[left + 1])
                        left++;
                    left++;
                }
                else{
```

Saved

```
14
15          int last = nums[0];
16          int left, right, sum, num;
17          int dist = abs(ans - target);
18          int i = 0;
19
20          for(; i < len - 2; i++){
21              if (i && nums[i] == last)
22                  continue;
23              last = num = nums[i];
24              left = i + 1;
25              right = len - 1;
26              while(left < right){
27                  sum = num + nums[left] + nums[right];
28                  if (sum == target) return sum;
29                  if(abs(sum - target) < dist){
30                      ans = sum;
31                      dist = abs(ans - target);
32                  }
33                  if(sum < target){
34                      while(left < right && nums[left] == nums[left + 1])
35                          left++;
36                      left++;
37                  }
38                  else{
39                      while(left < right && nums[right] == nums[right - 1])
40                          right--;
41                      right--;
42                  }
43              }
44          }
45          return ans;
46      }
47  };
```

```
14
15          int last = nums[0];
16          int left, right, sum, num;
17          int dist = abs(ans - target);
18          int i = 0;
19
20          for(; i < len - 2; i++){
21              if (i && nums[i] == last)
22                  continue;
23              last = num = nums[i];
24              left = i + 1;
25              right = len - 1;
26              while(left < right){
27                  sum = num + nums[left] + nums[right];
28                  if (sum == target) return sum;
```

Saved

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =

[-1,2,1,-4]

target =

1

Output

2

```cpp
class Solution {
public:
    string map[10] = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
    void helper(string digits, vector<string>& ans, int index, string current) {
        if(index == digits.length()) {
            ans.push_back(current);
            return;
        }
        string s = map[digits[index]-'0'];
        for(int i = 0; i < s.length(); i++) {
            helper(digits, ans, index+1, current+s[i]);
        }
    }
    vector<string> letterCombinations(string digits) {
        vector<string> ans;

        if(digits.length() == 0) return ans;

        helper(digits, ans, 0, "");
        return ans;
    }
};
```

```
    vector<string> letterCombinations(string digits) {
        vector<string> ans;

        if(digits.length() == 0) return ans;

        helper(digits, ans, 0, "");
        return ans;
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 3 ms

• Case 1 • Case 2 • Case 3

Input

digits =

"23"

Output

["ad","ae","af","bd","be","bf","cd","ce","cf"]

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        if(nums.size() < 4)
            return {};

        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());

        for(int i = 0; i < nums.size() - 3; increment(nums, i)){
            for(int j = i + 1; j < nums.size() - 2; increment(nums, j)){
                int l = j + 1, r = nums.size() - 1;
                long sum = 0;

                // two pointer
                while(l < r){
                    sum = (long)nums[i] + nums[j] + nums[l] + nums[r];

                    // if a solution is found, add it and change both pointers
                    if(sum == target){
                        ans.push_back({nums[i], nums[j], nums[l], nums[r]});
                        increment(nums, l);
                        decrement(nums, r);
                    }
                    // if sum is lesser, move left pointer to right by 1
                    else if(sum < target)
                        increment(nums, l);
                    // else if sum is greater, move right pointer to left by 1
                    else
                        decrement(nums, r);
                }
            }
        }

        return ans;
    }

    // increment() & decrement() are helper functions to avoid duplicates
```

Saved

```
                // if a solution is found, add it and change both pointers
                if(sum == target){
                    ans.push_back({nums[i], nums[j], nums[l], nums[r]});
                    increment(nums, l);
                    decrement(nums, r);
                }
                // if sum is lesser, move left pointer to right by 1
                else if(sum < target)
                    increment(nums, l);
                // else if sum is greater, move right pointer to left by 1
                else
                    decrement(nums, r);
            }
        }
    }

    return ans;
}

// increment() & decrement() are helper functions to avoid duplicates
// basically, they do i++ or i-- but until a different number is found

void increment(vector<int>& nums, int& n){
    do
        n++;
    while(n < nums.size() && nums[n] == nums[n - 1]);
}

void decrement(vector<int>& nums, int& n){
    do
        n--;
    while(n >= 0 && nums[n] == nums[n + 1]);
}
};
```

```
19                // if a solution is found, add it and change both pointers
20                if(sum == target){
21                    ans.push_back({nums[i], nums[j], nums[l], nums[r]});
22                    increment(nums, l);
23                    decrement(nums, r);
24                }
```

Saved

Testcase | Test Result

• Case 1 • Case 2

Input

nums =

[1,0,-1,0,-2,2]

target =

0

Output

[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

```cpp
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        // if list.size == 1 and position want to delete is 1 => NULL
        if(n == 1 && head->next==NULL) return NULL;

        // find size of list
        int count = 0;
        ListNode* temp = head;
        while(temp != NULL)
        {
            count++;
            temp = temp -> next;
        }

        // calculate position of node need to delete from head
        int nFromHead = count - n;

        // find the prev node of node want to delete
        temp = head;
        for(int i = 1 ; i < nFromHead ; i++)
        {
            temp = temp->next;
        }

        // nFromHead == 0 mean the first node need to delete
        // => head == head -> next

        // nFromHead != 0 => temp is prev node of node need to delete
        // => temp->next->next is the next node of node need to delete
        // => temp->next = temp->next->next
        (nFromHead != 0) ? temp->next = temp->next->next : head=head->next;
        return head;
    }
};
```

```cpp
class Solution {
public:
    int maxArea(vector<int>& height) {
        int left = 0;
        int right = height.size() - 1;
        int maxArea = 0;

        while (left < right) {
            int currentArea = min(height[left], height[right]) * (right - left);
            maxArea = max(maxArea, currentArea);

            if (height[left] < height[right]) {
                left++;
            } else {
                right--;
            }
        }

        return maxArea;
    }
};
```


Testcase | Test Result

Case 1 Case 2 +

height =

```
[1,8,6,2,5,4,8,3,7]
```

```cpp
//Approach 1 :
//time complexity - O(1) since the algorithm always iterates through a constant number of values (13 in this case).
//O(1) since the amount of extra space used is constant (the size of the storeIntRoman vector, which is also 13 in this case
class Solution {
public:
    string intToRoman(int num) {
        string Roman = "";
        // Creating vector of pairs to store the Roman numeral values and their corresponding symbols
        vector<pair<int, string>> storeIntRoman = {{1000, "M"}, {900, "CM"}, {500, "D"}, {400, "CD"}, {100, "C"}, {90, "XC"}, {50, "L"}, {40, "XL"}, {10, "X"}, {9, "IX"}, {5, "V"}, {4, "IV"}, {1, "I"}};
        // Iterating through the vector and repeatedly adds the corresponding symbols to the result string while subtracting the corresponding value from the input integer until the input integer becomes zero.
        for (int i = 0; i < storeIntRoman.size(); i++) {
            while (num >= storeIntRoman[i].first) {
                Roman += storeIntRoman[i].second;
                num -= storeIntRoman[i].first;
            }
        }
        return Roman;
    }
};
```

Saved

Ln 19, C

Testcase | Test Result

Accepted Runtime: 2 ms

• Case 1 • Case 2 • Case 3

Input

num =

3749

Output

"MMMDCCXLIX"

```cpp
class Solution {
public:
    int romanToInt(string s) {
        unordered_map<char, int> m;

        m['I'] = 1;
        m['V'] = 5;
        m['X'] = 10;
        m['L'] = 50;
        m['C'] = 100;
        m['D'] = 500;
        m['M'] = 1000;

        int ans = 0;

        for(int i = 0; i < s.length(); i++){
            if(m[s[i]] < m[s[i+1]]){
                ans -= m[s[i]];
            }
            else{
                ans += m[s[i]];
            }
        }
        return ans;
    }
};
```

```
        m['I'] = 1;
        m['V'] = 5;
        m['X'] = 10;
        m['L'] = 50;
        m['C'] = 100;
        m['D'] = 500;
        m['M'] = 1000;

        int ans = 0;

        for(int i = 0; i < s.length(); i++){
            if(m[s[i]] < m[s[i+1]]){
                ans -= m[s[i]];
            }
            else{
                ans += m[s[i]];
            }
        }
        return ans;
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 4 ms

• Case 1 • Case 2 • Case 3

Input

s =

"III"

Output

3

```cpp
class Solution {
public:
    string longestCommonPrefix(vector<string>& v) {
        string ans="";
        sort(v.begin(),v.end());
        int n=v.size();
        string first=v[0],last=v[n-1];
        for(int i=0;i<min(first.size(),last.size());i++){
            if(first[i]!=last[i]){
                return ans;
            }
            ans+=first[i];
        }
        return ans;
    }
};
```

Saved

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

strs =

```
["flower","flow","flight"]
```

Output

```
"fl"
```

```cpp
//Optimized Approach - O(n^2 logn + nlogn) - o(n^2 logn) time and O(n) space
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        int target = 0;
        sort(nums.begin(), nums.end());
        set<vector<int>> s;
        vector<vector<int>> output;
        for (int i = 0; i < nums.size(); i++){
            int j = i + 1;
            int k = nums.size() - 1;
            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == target) {
                    s.insert({nums[i], nums[j], nums[k]});
                    j++;
                    k--;
                } else if (sum < target) {
                    j++;
                } else {
                    k--;
                }
            }
        }
        for(auto triplets : s)
            output.push_back(triplets);
        return output;
    }
};
```

```cpp
        sort(nums.begin(), nums.end());
        set<vector<int>> s;
        vector<vector<int>> output;
        for (int i = 0; i < nums.size(); i++){
            int j = i + 1;
            int k = nums.size() - 1;
            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == target) {
                    s.insert({nums[i], nums[j], nums[k]});
                    j++;
                    k--;
                } else if (sum < target) {
                    j++;
                } else {
                    k--;
                }
            }
        }
        for(auto triplets : s)
            output.push_back(triplets);
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =

[-1,0,1,2,-1,-4]

Output

[[-1,-1,2],[-1,0,1]]

```cpp
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        int len = nums.size();
        if(len == 3) return nums[0] + nums[1] + nums[2];

        sort(nums.begin(), nums.end());

        int ans = nums[0] + nums[1] + nums[2];
        if(ans >= target) return ans;

        int max = nums[len - 1] + nums[len - 2] + nums[len - 3];
        if(max <= target) return max;

        int last = nums[0];
        int left, right, sum, num;
        int dist = abs(ans - target);
        int i = 0;

        for(; i < len - 2; i++){
            if (i && nums[i] == last)
                continue;
            last = num = nums[i];
            left = i + 1;
            right = len - 1;
            while(left < right){
                sum = num + nums[left] + nums[right];
                if (sum == target) return sum;
                if(abs(sum - target) < dist){
                    ans = sum;
                    dist = abs(ans - target);
                }
                if(sum < target){
                    while(left < right && nums[left] == nums[left + 1])
                        left++;
                    left++;
                }
                else{
```

Saved

```
14
15          int last = nums[0];
16          int left, right, sum, num;
17          int dist = abs(ans - target);
18          int i = 0;
19
20          for(; i < len - 2; i++){
21              if (i && nums[i] == last)
22                  continue;
23              last = num = nums[i];
24              left = i + 1;
25              right = len - 1;
26              while(left < right){
27                  sum = num + nums[left] + nums[right];
28                  if (sum == target) return sum;
29                  if(abs(sum - target) < dist){
30                      ans = sum;
31                      dist = abs(ans - target);
32                  }
33                  if(sum < target){
34                      while(left < right && nums[left] == nums[left + 1])
35                          left++;
36                      left++;
37                  }
38                  else{
39                      while(left < right && nums[right] == nums[right - 1])
40                          right--;
41                      right--;
42                  }
43              }
44          }
45          return ans;
46      }
47  };
```

```
14
15          int last = nums[0];
16          int left, right, sum, num;
17          int dist = abs(ans - target);
18          int i = 0;
19
20          for(; i < len - 2; i++){
21              if (i && nums[i] == last)
22                  continue;
23              last = num = nums[i];
24              left = i + 1;
25              right = len - 1;
26              while(left < right){
27                  sum = num + nums[left] + nums[right];
28                  if (sum == target) return sum;
```

Saved

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =

[-1,2,1,-4]

target =

1

Output

2

```cpp
class Solution {
public:
    string map[10] = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
    void helper(string digits, vector<string>& ans, int index, string current) {
        if(index == digits.length()) {
            ans.push_back(current);
            return;
        }
        string s = map[digits[index]-'0'];
        for(int i = 0; i < s.length(); i++) {
            helper(digits, ans, index+1, current+s[i]);
        }
    }
    vector<string> letterCombinations(string digits) {
        vector<string> ans;

        if(digits.length() == 0) return ans;

        helper(digits, ans, 0, "");
        return ans;
    }
};
```

```
    vector<string> letterCombinations(string digits) {
        vector<string> ans;

        if(digits.length() == 0) return ans;

        helper(digits, ans, 0, "");
        return ans;
    }
};
```

Saved

Testcase | Test Result

Accepted Runtime: 3 ms

• Case 1 • Case 2 • Case 3

Input

digits =

"23"

Output

["ad","ae","af","bd","be","bf","cd","ce","cf"]

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        if(nums.size() < 4)
            return {};

        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());

        for(int i = 0; i < nums.size() - 3; increment(nums, i)){
            for(int j = i + 1; j < nums.size() - 2; increment(nums, j)){
                int l = j + 1, r = nums.size() - 1;
                long sum = 0;

                // two pointer
                while(l < r){
                    sum = (long)nums[i] + nums[j] + nums[l] + nums[r];

                    // if a solution is found, add it and change both pointers
                    if(sum == target){
                        ans.push_back({nums[i], nums[j], nums[l], nums[r]});
                        increment(nums, l);
                        decrement(nums, r);
                    }
                    // if sum is lesser, move left pointer to right by 1
                    else if(sum < target)
                        increment(nums, l);
                    // else if sum is greater, move right pointer to left by 1
                    else
                        decrement(nums, r);
                }
            }
        }

        return ans;
    }

    // increment() & decrement() are helper functions to avoid duplicates
```

Saved

```
                // if a solution is found, add it and change both pointers
                if(sum == target){
                    ans.push_back({nums[i], nums[j], nums[l], nums[r]});
                    increment(nums, l);
                    decrement(nums, r);
                }
                // if sum is lesser, move left pointer to right by 1
                else if(sum < target)
                    increment(nums, l);
                // else if sum is greater, move right pointer to left by 1
                else
                    decrement(nums, r);
            }
        }
    }

    return ans;
}

// increment() & decrement() are helper functions to avoid duplicates
// basically, they do i++ or i-- but until a different number is found

void increment(vector<int>& nums, int& n){
    do
        n++;
    while(n < nums.size() && nums[n] == nums[n - 1]);
}

void decrement(vector<int>& nums, int& n){
    do
        n--;
    while(n >= 0 && nums[n] == nums[n + 1]);
}
};
```

```
19                // if a solution is found, add it and change both pointers
20                if(sum == target){
21                    ans.push_back({nums[i], nums[j], nums[l], nums[r]});
22                    increment(nums, l);
23                    decrement(nums, r);
24                }
```

Saved

Testcase | Test Result

• Case 1 • Case 2

Input

nums =

[1,0,-1,0,-2,2]

target =

0

Output

[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

```cpp
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        // if list.size == 1 and position want to delete is 1 => NULL
        if(n == 1 && head->next==NULL) return NULL;

        // find size of list
        int count = 0;
        ListNode* temp = head;
        while(temp != NULL)
        {
            count++;
            temp = temp -> next;
        }

        // calculate position of node need to delete from head
        int nFromHead = count - n;

        // find the prev node of node want to delete
        temp = head;
        for(int i = 1 ; i < nFromHead ; i++)
        {
            temp = temp->next;
        }

        // nFromHead == 0 mean the first node need to delete
        // => head == head -> next

        // nFromHead != 0 => temp is prev node of node need to delete
        // => temp->next->next is the next node of node need to delete
        // => temp->next = temp->next->next
        (nFromHead != 0) ? temp->next = temp->next->next : head=head->next;
        return head;
    }
};
```

```cpp
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        // if list.size == 1 and position want to delete is 1 => NULL
        if(n == 1 && head->next==NULL) return NULL;

        // find size of list
        int count = 0;
        ListNode* temp = head;
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

head =

[1,2,3,4,5]

n =

2

Output

[1,2,3,5]

```cpp
class Solution {
public:
    bool isValid(string s) {
        stack<char> st; // create an empty stack to store opening brackets
        for (char c : s) { // loop through each character in the string
            if (c == '(' || c == '{' || c == '[') { // if the character is an opening bracket
                st.push(c); // push it onto the stack
            } else { // if the character is a closing bracket
                if (st.empty() || // if the stack is empty or
                    (c == ')' && st.top() != '(') || // the closing bracket doesn't match the corresponding opening bracket at the top of the stack
                    (c == '}' && st.top() != '{') ||
                    (c == ']' && st.top() != '[')) {
                    return false; // the string is not valid, so return false
                }
                st.pop(); // otherwise, pop the opening bracket from the stack
            }
        }
        return st.empty(); // if the stack is empty, all opening brackets have been matched with their corresponding closing brackets,
                           // so the string is valid, otherwise, there are unmatched opening brackets, so return false
    }
};
```

```
        return st.empty(); // if the stack is empty, all opening brackets have been matched with their corresponding closing brackets,
19                         // so the string is valid, otherwise, there are unmatched opening brackets, so return false
20      }
21  };
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

s =

"()"

Output

true