11. Container With Most Water

C++ solution:

```cpp
class Solution {
public:
    int maxArea(vector<int>& height) {
        int n = height.size();
        int max_volume = 0, current_volume = 0;
        for(int i = 0; i < n - 1; ++i)
        {
            for(int j = i + 1; j < n; ++j)
            {
                current_volume = (j - i) * min(height[i], height[j]);
                if (current_volume > max_volume)
                {
                    max_volume = current_volume;
                }
            }
        }
        return max_volume;
    }
};
```

12. Integer to Roman

C++ solution:

```cpp
class Solution {
public:
    string intToRoman(int num) {
        string result;

        int current_digit = num / 1000;
        int num_left = num % 1000;
        for(int i = 0; i < current_digit; ++i)
        {
            result += 'M';
        }

        current_digit = num_left / 100;
        num_left %= 100;
        if (current_digit == 4)
        {
            result += "CD";
        }
        else if (current_digit == 9)
        {
            result += "CM";
        }
        else
        {
            if (current_digit > 4)
            {
                result += 'D';
                current_digit -= 5;
            }
            for(int i = 0; i < current_digit; ++i)
            {
                result += 'C';
            }
        }

        current_digit = num_left / 10;
        num_left %= 10;
        if (current_digit == 4)
        {
            result += "XL";
        }
        else if (current_digit == 9)
        {
            result += "XC";
        }
        else
        {
            if (current_digit > 4)
            {
                result += 'L';
                current_digit -= 5;
            }
            for(int i = 0; i < current_digit; ++i)
            {
                result += 'X';
            }
        }

        current_digit = num_left;
        if (current_digit == 4)
        {
            result += "IV";
        }
        else if (current_digit == 9)
        {
            result += "IX";
        }
        else
        {
            if (current_digit > 4)
            {
                result += 'V';
                current_digit -= 5;
            }
            for(int i = 0; i < current_digit; ++i)
            {
                result += 'I';
            }
        }

        return result;
    }
};
```

[13. Roman to Integer]

C++ solution:

```
class Solution {
public:
    int romanToInt(string s) {
        int length = s.length();
        int result = 0;

        for(int i = 0; i < length; ++i)
        {
            if (s[i] == 'M')
            {
                result += 1000;
            }
            else if (s[i] == 'D')
            {
                result += 500;
            }
            else if (s[i] == 'C')
            {
                if (i < length - 1 && s[i + 1] == 'D')
                {
                    result += 400;
                    ++i;
                }
                else if (i < length - 1 && s[i + 1] == 'M')
                {
                    result += 900;
                    ++i;
                }
                else
                {
                    result += 100;
                }
            }
            else if (s[i] == 'L')
            {
                result += 50;
            }
            else if (s[i] == 'X')
            {
                if (i < length - 1 && s[i + 1] == 'L')
                {
                    result += 40;
                    ++i;
                }
                else if (i < length - 1 && s[i + 1] == 'C')
                {
                    result += 90;
                    ++i;
                }
                else
                {
                    result += 10;
                }
            }
            else if (s[i] == 'V')
            {
                result += 5;
            }
            else if (s[i] == 'I')
            {
                if (i < length - 1 && s[i + 1] == 'V')
                {
                    result += 4;
                    ++i;
                }
                else if (i < length - 1 && s[i + 1] == 'X')
                {
                    result += 9;
                    ++i;
                }
                else
                {
                    result += 1;
                }
            }
        }
        return result;
    }
};
```

14. Longest Common Prefix

C++ solution:

```
class Solution {
public:
    string longestCommonPrefix(vector<string>& strs) {
        int result = 0;
        int num = strs.size();

        if (num == 0)
        {
            return "";
        }

        if (num == 1)
        {
            return strs[0];
        }

        int min_length = strs[0].length();
        for(int i = 1; i < num; ++i)
        {
            if (strs[i].length() < min_length)
            {
                min_length = strs[i].length();
            }
        }

        while (true)
        {
            if (result >= min_length)
            {
                break;
            }

            int i = 1;
            for (; i < num; ++i)
            {
                if (strs[i][result] != strs[i - 1][result])
                {
                    break;
                }
            }
            if (i < num)
            {
                break;
            }
            ++result;
        }

        return strs[0].substr(0, result);
    }
};
```

15. 3Sum

C++ solution:

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        vector<vector<int>> result;

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

        int n = nums.size();
        int left = 0, right = 0, sum = 0;

        for (int i = 0; i < n - 2; ++i)
        {
            if (i > 0 && nums[i] == nums[i - 1])
            {
                continue;
            }

            if (nums[i] + nums[i + 1] + nums[i + 2] > 0)
            {
                break;
            }

            if (nums[i] + nums[n - 2] + nums[n - 1] < 0)
            {
                continue;
            }

            left = i + 1, right = n - 1;

            while (left < right)
            {
                sum = nums[i] + nums[left] + nums[right];
                if (sum == 0)
                {
                    vector<int> temp = { nums[i], nums[left], nums[right] };
                    if (result.empty() || (*(result.end() - 1) != temp))
                    {
                        result.insert(result.end(), temp);
                    }
                    ++left;
                    --right;
                }
                else if (sum < 0)
                {
                    ++left;
                }
                else
                {
                    --right;
                }
            }
        }

        return result;
    }
};
```

[16. 3Sum Closest]

C++ solution:

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {

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

        int n = nums.size();
        int sum = 0;
        int result = nums[0] + nums[1] + nums[2];

        for (int i = 0; i < n - 2; ++i)
        {
            sum = nums[i] + nums[i + 1] + nums[i + 2];
            if (sum >= target)
            {
                if (sum - target < abs(result - target))
                {
                    result = sum;
                }
                break;
            }

            for (int j = i + 1; j < n - 1; ++j)
            {
                sum = nums[i] + nums[j] + nums[j + 1];
                if (sum >= target)
                {
                    if (sum - target < abs(result - target))
                    {
                        result = sum;
                    }
                    break;
                }

                for (int k = j + 1; k < n; ++k)
                {
                    sum = nums[i] + nums[j] + nums[k];

                    if (sum >= target)
                    {
                        if (sum - target < abs(result - target))
                        {
                            result = sum;
                        }

                        sum = nums[i] + nums[j] + nums[k - 1];
                        if (k > j + 1 && target - sum < abs(result - target))
                        {
                            result = sum;
                        }
                        break;
                    }

                    if (k == n - 1)
                    {
                        if (target - sum < abs(result - target))
                        {
                            result = sum;
                        }
                    }

                }
            }
        }

        return result;
```

```
    }
};
```

[17. Letter Combinations of a Phone Number]

C++ solution:

```
class Solution {
public:
    vector<string> letterCombinations(string digits) {

        unordered_map<char, set<char>> number_letters;
        number_letters['2'] = set<char>{ 'a', 'b', 'c' };
        number_letters['3'] = set<char>{ 'd', 'e', 'f' };
        number_letters['4'] = set<char>{ 'g', 'h', 'i' };
        number_letters['5'] = set<char>{ 'j', 'k', 'l' };
        number_letters['6'] = set<char>{ 'm', 'n', 'o' };
        number_letters['7'] = set<char>{ 'p', 'q', 'r', 's' };
        number_letters['8'] = set<char>{ 't', 'u', 'v' };
        number_letters['9'] = set<char>{ 'w', 'x', 'y', 'z' };

        vector<string> results;

        int n = digits.length();

        if (n > 1)
        {
            for (string s : letterCombinations(digits.substr(1)))
            {
                for(char c : number_letters[digits[0]])
                {
                    string current_s = c + s;
                    results.insert(results.end(), current_s);
                }

            }
        }
        else
        {
            for(char c : number_letters[digits[0]])
            {
                stringstream stream;
                stream << c;
                string current_s = stream.str();
                results.insert(results.end(), current_s);
            }
        }

        return results;
    }
};
```

18. 4Sum

C++ solution:

```cpp
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> results;

        sort(nums.begin(), nums.end());
        int n = nums.size(), curSum = 0;

        if (n < 4 || nums[n - 4] + nums[n - 3] + nums[n - 2] + nums[n - 1] < target)
        {
            return results;
        }

        for (int i = 0; i < n - 3; ++i)
        {
            if (i > 0 && nums[i] == nums[i - 1])
            {
                continue;
            }

            if (nums[i] + nums[i + 1] + nums[i + 2] + nums[i + 3] > target)
            {
                break;
            }

            if (nums[i] + nums[n - 3] + nums[n - 2] + nums[n - 1] < target)
            {
                continue;
            }

            for(int j = i + 1; j < n - 2; ++j)
            {
                if (j > i + 1 && nums[j] == nums[j - 1])
                {
                    continue;
                }

                if (nums[i] + nums[j] + nums[j + 1] + nums[j + 2] > target)
                {
                    break;
                }

                if (nums[i] + nums[j] + nums[n - 2] + nums[n - 1] < target)
                {
                    continue;
                }
                for (int k = j + 1; k < n - 1; ++k)
                {
                    if (k > j + 1 && nums[k] == nums[k - 1])
                    {
                        continue;
                    }

                    if (nums[i] + nums[j] + nums[k] + nums[k + 1] > target)
                    {
                        break;
                    }

                    if (nums[i] + nums[j] + nums[k] + nums[n - 1] < target)
                    {
                        continue;
                    }

                    for(int m = k + 1; m < n; ++m)
                    {
                        if (m > k + 1 && nums[m] == nums[m - 1])
                        {
                            continue;
                        }
                        curSum = nums[i] + nums[j] + nums[k] + nums[m];
                        if (curSum > target)
                        {
                            break;
                        }

                        if (curSum == target)
                        {
                            results.insert(results.end(), { nums[i], nums[j], nums[k],
 nums[m] });
                            break;
                        }
                    }
                }
            }
        }

        return results;
    }
};
```

19. Remove Nth Node From End of List

C++ solution:

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        ListNode *first = head, *second = head;

        int count = 0;

        if (!head->next)
        {
            return nullptr;
        }

        while (count < n && first->next)
        {
            ++count;
            first = first->next;
        }

        if (count < n)
        {
            return head->next;
        }

        while (first->next)
        {
            first = first->next;
            second = second->next;
        }

        second->next = second->next->next;

        return head;
    }
};
```

20. Valid Parentheses

C++ solution:

```cpp
class Solution {
public:
    bool isValid(string s) {
        stack<char> brackets;
        for(char c : s)
        {
            if (c == '(' || c == '{' || c == '[')
            {
                brackets.push(c);
            }
            else if (brackets.empty())
            {
                return false;
            }
            else if ((c == ')' && brackets.top() == '(') ||
                (c == '}' && brackets.top() == '{') ||
                (c == ']' && brackets.top() == '['))
            {
                brackets.pop();
            }
            else
            {
                return false;
            }
        }

        if (brackets.empty())
        {
            return true;
        }
        return false;
    }
};
```