Problem 1: Word Match with Underscore

```
#include <iostream>
#include <sstream>
#include <vector>
#include <algorithm>
using namespace std;

string matchFound(string input1, string input2) {
    stringstream ss(input2);
    string word, result = "";
    for (int i = 0; i < input2.size(); ++i)
        input2[i] = tolower(input2[i]);

    while (getline(ss, word, ':')) {
        if (word.length() != input1.length()) continue;
        bool match = true;
        for (int i = 0; i < word.length(); i++) {
            if (input1[i] != '_' && tolower(input1[i]) != word[i]) {
                match = false;
                break;
            }
        }
        if (match) {
            transform(word.begin(), word.end(), word.begin(), ::toupper);
            if (!result.empty()) result += ":";
            result += word;
        }
    }
    return result;
}
```

Problem 4: Find the Added Character

```
#include <iostream>
using namespace std;

char findAddedChar(string s, string t) {
    char res = 0;
    for (char c : s) res ^= c;
    for (char c : t) res ^= c;
    return res;
}
```

Problem 5: Next Greater Element

```
#include <iostream>
#include <vector>
#include <stack>
#include <unordered_map>
using namespace std;

vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
    stack<int> s;
    unordered_map<int, int> mp;
    for (int i = nums2.size() - 1; i >= 0; --i) {
        while (!s.empty() && s.top() <= nums2[i]) s.pop();
```

```
        mp[nums2[i]] = s.empty() ? -1 : s.top();
        s.push(nums2[i]);
    }
    vector<int> res;
    for (int num : nums1)
        res.push_back(mp[num]);
    return res;
}
```

Problem 6: Balanced Parentheses

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

bool isBalanced(string str) {
    stack<char> s;
    for (char c : str) {
        if (c == '(' || c == '[' || c == '{') s.push(c);
        else if (c == ')' && !s.empty() && s.top() == '(') s.pop();
        else if (c == ']' && !s.empty() && s.top() == '[') s.pop();
        else if (c == '}' && !s.empty() && s.top() == '{') s.pop();
        else return false;
    }
    return s.empty();
}
```

Problem 10: Find First and Last Position

```
#include <iostream>
#include <vector>
using namespace std;

int findIndex(vector<int>& nums, int target, bool findFirst) {
    int l = 0, r = nums.size() - 1, ans = -1;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (nums[mid] == target) {
            ans = mid;
            if (findFirst) r = mid - 1;
            else l = mid + 1;
        }
        else if (nums[mid] < target) l = mid + 1;
        else r = mid - 1;
    }
    return ans;
}

vector<int> searchRange(vector<int>& nums, int target) {
    return {findIndex(nums, target, true), findIndex(nums, target, false)};
}
```