Coding Solution

```cpp
#include <iostream>
#include <stack>
#include <queue>
#include <string>
#include <vector>
#include <map>

using namespace std;

// Q1: Valid Parentheses
bool isValid(string s) {
    stack<char> st;
    for (char c : s) {
        if (c == '(' || c == '{' || c == '[') {
            st.push(c);
        } else {
            if (st.empty()) return false;
            if ((c == ')' && st.top() != '(') ||
                (c == '}' && st.top() != '{') ||
                (c == ']' && st.top() != '[')) {
                return false;
            }
            st.pop();
        }
    }
    return st.empty();
}

// Q2: Next Greater Element (Non-Circular)
int nextGreaterElement(vector<int>& arr, int element) {
    for (size_t i = 0; i < arr.size(); ++i) {
        if (arr[i] == element) {
            for (size_t j = i + 1; j < arr.size(); ++j) {
                if (arr[j] > element) {
                    return j - i;
                }
            }
            return -1; // Not found
        }
    }
    return -1; // Element not in array
}
```

```
// Q3: Next Greater Element (Circular)
int nextGreaterElementCircular(vector<int>& arr, int element) {
    int n = arr.size();
    for (int i = 0; i < n; ++i) {
        if (arr[i] == element) {
            for (int j = i + 1; j < n; ++j) {
                if (arr[j] > element) return j - i;
            }
            for (int j = 0; j < i; ++j) {
                if (arr[j] > element) return j + (n-i);
            }
            return -1; // Not found
        }
    }
    return -1; // Element not in array

}


// Q4: First Non-Repeating Character
pair<char, int> firstNonRepeatingCharacter(string s) {
    queue<char> q;
    map<char, int> count;

    for (char c : s) {
        q.push(c);
        count[c]++;
    }

    while (!q.empty()) {
        char c = q.front();
        q.pop();
        if (count[c] == 1) {
            return make_pair(c, s.find(c));
        }
    }
    return make_pair(' ', -1); // No non-repeating character
}


int main() {
    // Q1 Test Cases
    cout << "Q1 Test Cases:\n";
    cout << isValid("()") << endl;      // true
    cout << isValid("()[]{}") << endl;   // true
    cout << isValid("(]") << endl;     // false
```

```
    cout << isValid("{[()]}") << endl; //true
    cout << isValid("([)]") << endl;    //false

    // Q2 Test Cases
    cout << "\nQ2 Test Cases:\n";
    vector<int> arr1 = {1, 4, 2, 5, 0, 6, 7};
    cout << nextGreaterElement(arr1, 4) << endl;  // 2
    cout << nextGreaterElement(arr1, 2) << endl;  // 1
    cout << nextGreaterElement(arr1, 7) << endl;  // -1 (Not Found)
    vector<int> arr2 = {10, 6, 7, 2, 5, 1, 0, 4};
    cout << nextGreaterElement(arr2, 7) << endl; // -1 (Not Found)

    // Q3 Test Cases
    cout << "\nQ3 Test Cases:\n";
    cout << nextGreaterElementCircular(arr1, 4) << endl;  // 2
    cout << nextGreaterElementCircular(arr1, 2) << endl;  // 1
    cout << nextGreaterElementCircular(arr1, 7) << endl;  // 1
    cout << nextGreaterElementCircular(arr2, 7) << endl;  // 6


    // Q4 Test Cases
    cout << "\nQ4 Test Cases:\n";
    pair<char, int> result1 = firstNonRepeatingCharacter("thisisDSlab");
    cout << "Character: " << result1.first << ", Index: " << result1.second << endl; // t, 0
    pair<char, int> result2 = firstNonRepeatingCharacter("CodeForDSlabClass");
    cout << "Character: " << result2.first << ", Index: " << result2.second << endl; // d, 2
    pair<char, int> result3 =
firstNonRepeatingCharacter("Thequickbrownfoxjumpsoveralazydog");
    cout << "Character: " << result3.first << ", Index: " << result3.second << endl; //  , -1

    return 0;
}
```