Coding Solution

```cpp
#include <iostream>
#include <stack>
#include <queue>
#include <string>
#include <unordered_map>

using namespace std;

// Q1: Valid Parentheses
bool isValidParentheses(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(int arr[], int n, int element) {
    stack<pair<int, int>> st; // {value, index}

    for (int i = 0; i < n; ++i) {
        while (!st.empty() && arr[st.top().second] < arr[i]) {
            if (st.top().first == element) {
                return i - st.top().second;
            }
            st.pop();
        }
        st.push({arr[i], i});
    }
    return -1; // Not found
}
```

```
// Q3: Next Greater Element (Circular)
int nextGreaterElementCircular(int arr[], int n, int element) {
    int index = -1;
    for (int i = 0; i < n; i++) {
        if (arr[i] == element) index = i;
    }
    if (index == -1) return -1;

    for (int i = index + 1; i < n; i++) {
        if (arr[i] > element) return i - index;
    }

    for (int i = 0; i < index; i++) {
        if (arr[i] > element) return n - index + i;
    }
    return -1; // Not found
}

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

    for (char c : s) {
        count[c]++;
        q.push(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); //none found
}


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


    // Q2 Test Cases
    cout << "\nQ2 Test Cases:" << endl;
    int arr1[] = {1, 4, 2, 5, 0, 6, 7};
    cout << nextGreaterElement(arr1, 7, 4) << endl;  // 2
    cout << nextGreaterElement(arr1, 7, 2) << endl;  // 1
    cout << nextGreaterElement(arr1, 7, 7) << endl;  // -1 (Not found)
    int arr2[] = {10, 6, 7, 2, 5, 1, 0, 4};
    cout << nextGreaterElement(arr2, 8, 7) << endl;  // -1 (Not found)


    // Q3 Test Cases
    cout << "\nQ3 Test Cases:" << endl;
    cout << nextGreaterElementCircular(arr1, 7, 4) << endl;  // 2
    cout << nextGreaterElementCircular(arr1, 7, 2) << endl;  // 1
    cout << nextGreaterElementCircular(arr1, 7, 7) << endl;  // 1
    cout << nextGreaterElementCircular(arr2, 8, 7) << endl;  // 6


    // Q4 Test Cases
    cout << "\nQ4 Test Cases:" << endl;
    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;
}
```