Coding Solution

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <limits> // for numeric_limits

using namespace std;

// Q1. Find the first pair of repeating elements
pair<int, int> findFirstRepeatingPair(const vector<int>& arr) {
    vector<int> seen;
    for (int i = 0; i < arr.size(); ++i) {
        for (int j = i + 1; j < arr.size(); ++j) {
            if (arr[i] == arr[j]) {
                return make_pair(arr[i], arr[j]); //Return the pair
            }
        }
    }
    return make_pair(-1, -1); // Return (-1,-1) if no repeating pair found.
}


// Q2. Maximum sum of two elements closest to zero
int maxSumClosestToZero(const vector<int>& arr) {
    int n = arr.size();
    if (n < 2) return 0; //Not enough elements

    int minSum = numeric_limits<int>::max(); // Initialize with maximum possible int value

    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            int currentSum = arr[i] + arr[j];
            if (abs(currentSum) < abs(minSum)) {
                minSum = currentSum;
            }
        }
    }
    return minSum;
}

// Q3. Find the missing element in an Arithmetic Progression
int findMissingAPval(const vector<int>& arr) {
    if (arr.size() < 2) return -1; //Not enough elements for AP
```

```
    int diff = arr[1] - arr[0];
    for (size_t i = 2; i < arr.size(); ++i) {
        if (arr[i] - arr[i - 1] != diff) {
            return arr[i - 1] + diff;
        }
    }
    return -1; // No missing element found.
}


// Q4. Find first and last occurrences of an element in a sorted array
pair<int, int> findFirstLastOccurrence(const vector<int>& arr, int x) {
    int first = -1, last = -1;
    for (int i = 0; i < arr.size(); ++i) {
        if (arr[i] == x) {
            if (first == -1) first = i + 1; //Adjusting for 1-based indexing
            last = i + 1; //Adjusting for 1-based indexing
        }
    }
    return make_pair(first, last);
}

// Q5. Interpolation search
int interpolationSearch(const vector<int>& arr, int k) {
    int low = 0, high = arr.size() - 1;
    while (low <= high && k >= arr[low] && k <= arr[high]) {
        int pos = low + ((double)(high - low) / (arr[high] - arr[low])) * (k - arr[low]);
        if (arr[pos] == k) return pos + 1; //Adjusting for 1-based indexing
        else if (arr[pos] < k) low = pos + 1;
        else high = pos - 1;
    }
    return -1; //Not found
}

// Q6. Maximum length subarray with first element >= last element
int maxLengthSubarray(const vector<int>& arr) {
    int maxLen = 0;
    for (int i = 0; i < arr.size(); ++i) {
        for (int j = i; j < arr.size(); ++j) {
            if (arr[i] >= arr[j]) {
                maxLen = max(maxLen, j - i + 1);
            }
        }
    }
    return maxLen;
}
```

```
// Q7. Find index of a string in an array of strings
int findStringIndex(const vector<string>& arr, const string& x) {
    for (size_t i = 0; i < arr.size(); ++i) {
        if (arr[i] == x) return i + 1; //Adjusting for 1-based indexing
    }
    return 0; //Not found
}


// Q8. Linear search for an element
bool linearSearch(const vector<int>& arr, int x) {
    for (int val : arr) {
        if (val == x) return true;
    }
    return false;
}


int main() {
    // Example usage (replace with your own test cases)
    vector<int> arr1 = {3, 6, 12, -10, 3, 3, 6, 34, 0, -109, 98, 1};
    vector<int> arr2 = {-5, -50, 56};
    vector<int> arr3 = {2, 4, 8, 12, 16, 22};
    vector<int> arr4 = {1, 2, 2, 3, 3, 3, 4, 4, 4, 4};
    vector<int> arr6 = {-5, -1, 7, 5, 1, -2};
    vector<string> arr7 = {"Hi", "Folks", "ide", "for", "practice"};

    cout << "Q1: First repeating pair: " << findFirstRepeatingPair(arr1).first << ", " <<
findFirstRepeatingPair(arr1).second << endl;
    cout << "Q2: Max sum closest to zero: " << maxSumClosestToZero(arr2) << endl;
    cout << "Q3: Missing AP element: " << findMissingAPval(arr3) << endl;
    cout << "Q4: First and last occurrence of 3: " << findFirstLastOccurrence(arr4, 3).first
<< ", " << findFirstLastOccurrence(arr4, 3).second << endl;
    cout << "Q5: Position of 7 in {2,4,7,9,10}: " << interpolationSearch({2,4,7,9,10}, 7) <<
endl;
    cout << "Q6: Max length subarray: " << maxLengthSubarray(arr6) << endl;
    cout << "Q7: Index of \"ide\": " << findStringIndex(arr7, "ide") << endl;
    cout << "Q8: Is 5 present in {1,2,3,4,5}: " << linearSearch({1,2,3,4,5}, 5) << endl;

    return 0;
}
```

Remember to save this code as `EnrollNo_W3_LabA_QuestionNo.cpp` (replace `EnrollNo` and `QuestionNo` accordingly) and compile it using a C++ compiler (like g+

+). The output will be printed to the console. Make sure to handle potential errors (like empty arrays or invalid inputs) more robustly in a production environment. The current code provides a basic solution for each problem.