Coding Solution

This response provides solutions for questions 1, 2, 3, and 4. Due to the length of the assignment, the remaining questions are best addressed separately. Remember to save your files using the specified nomenclature (EnrollNo_W3_LabA_QuestionNo.cpp).

Q1. Find the first pair of repeating elements

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>

using namespace std;

int findFirstRepeatingPair(const vector<int>& arr) {
    unordered_map<int, int> count;
    for (int i = 0; i < arr.size(); ++i) {
        if (++count[arr[i]] == 2) {
            return i + 1; // Adjust to 1-based indexing
        }
    }
    return 0; // No repeating pair found
}

int main() {
    vector<int> arr = {3, 6, 12, -10, 3, 3, 6, 34, 0, -109, 98, 1};
    int result = findFirstRepeatingPair(arr);
    if (result) {
        cout << "First repeating pair found at index: " << result << endl;
    } else {
        cout << "No repeating pair found." << endl;
    }
    return 0;
}
```

Q2. Maximum sum of two elements closest to zero

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

```
using namespace std;

int maxSumClosestToZero(vector<int>& arr) {
    int n = arr.size();
    if (n < 2) return 0; // Not enough elements for a pair

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

    int min_sum = numeric_limits<int>::max(); // Initialize to largest possible integer
    int left = 0;
    int right = n - 1;

    while (left < right) {
        int current_sum = arr[left] + arr[right];
        if (abs(current_sum) < abs(min_sum)) {
            min_sum = current_sum;
        }
        if (current_sum < 0) {
            left++;
        } else {
            right--;
        }
    }
    return min_sum;
}

int main() {
    vector<int> arr = {-5, -50, 56};
    int result = maxSumClosestToZero(arr);
    cout << "Maximum sum of two elements closest to zero: " << result << endl;
    return 0;
}
```

Q3. Find the missing element in an Arithmetic Progression

```cpp
#include <iostream>
#include <vector>

using namespace std;

int findMissingAPval(const vector<int>& arr) {
    if (arr.size() < 2) return 0; // Not enough elements to form an 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 0; // No missing element found (or not an AP)

}

int main() {
    vector<int> arr = {2, 4, 8, 10, 12}; //Example with 6 missing
    int missing = findMissingAPval(arr);
    if (missing !=0){
        cout << "Missing element in AP: " << missing << endl;
    } else {
        cout << "No missing element found or not an AP." << endl;
    }
    return 0;
}
```

Q4. Find the first and last occurrences of an element in a sorted array

```cpp
#include <iostream>
#include <vector>

using namespace std;

pair<int, int> findFirstAndLast(const vector<int>& arr, int x) {
    int n = arr.size();
    int first = -1, last = -1;

    // Find first occurrence using binary search
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            first = mid;
            right = mid - 1; // Continue searching on the left
        } else if (arr[mid] < x) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    // Find last occurrence using binary search
    left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            last = mid;
            left = mid + 1; // Continue searching on the right
        } else if (arr[mid] < x) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return make_pair(first +1, last +1); //Adjust to 1 based indexing

}

int main() {
    vector<int> arr = {2, 5, 5, 5, 6, 6, 8, 9, 9, 9};
    int x = 5;
    pair<int, int> result = findFirstAndLast(arr, x);
    if (result.first != -1) {
        cout << "First occurrence at index: " << result.first << endl;
        cout << "Last occurrence at index: " << result.second << endl;
    } else {
        cout << "Element not found." << endl;
    }
    return 0;
}
```

Remember to compile and run these codes using a C++ compiler (like g++). Replace `"EnrollNo_W3_LabA_QuestionNo.cpp"` with your actual enrollment number and question number. For example, if your enrollment number is 12345 and this is question 1, the filename should be `12345_W3_LabA_Question1.cpp`.