Arrays Problems

Problem 1: Largest element

Input:

```
n
a1 a2 ... an
```

Output largest element.

Method

Keep `maxVal = first element`. Traverse array and update if current element is larger.

C++ solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    int maxVal = a[0];
    for (int i = 1; i < n; i++) {
        if (a[i] > maxVal) maxVal = a[i];
    }
    cout << maxVal;
    return 0;
}
```

Problem 2: Second largest distinct element

Method

Maintain two values:

- largest
- second largest

Skip duplicates of largest.

C++ solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    int largest = INT_MIN, second = INT_MIN;

    for (int x : a) {
        if (x > largest) {
            second = largest;
            largest = x;
        } else if (x > second && x != largest) {
            second = x;
        }
    }

    if (second == INT_MIN) cout << "No second largest";
    else cout << second;
    return 0;
}
```

Problem 3: Reverse array

C++ solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    int l = 0, r = n - 1;
    while (l < r) {
        swap(a[l], a[r]);
        l++;
        r--;
    }

    for (int x : a) cout << x << " ";
```

```
    return 0;
}
```

Problem 4: Frequency of elements

C++ solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    unordered_map<int, int> freq;

    for (int i = 0; i < n; i++) {
        cin >> a[i];
        freq[a[i]]++;
    }

    for (auto &p : freq) {
        cout << p.first << " " << p.second << "\n";
    }
    return 0;
}
```

Problem 5: Pair with given sum

Method

Use hash set. For each x, check if `target - x` was seen.

C++ solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, target;
    cin >> n >> target;
    vector<int> a(n);
    unordered_set<int> seen;

    for (int i = 0; i < n; i++) {
        cin >> a[i];
        int need = target - a[i];
```

```
        if (seen.count(need)) {
            cout << "YES";
            return 0;
        }
        seen.insert(a[i]);
    }

    cout << "NO";
    return 0;
}
```

Problem 6: Maximum subarray sum

Kadane’s algorithm.

C++ solution

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    int best = a[0], current = a[0];
    for (int i = 1; i < n; i++) {
        current = max(a[i], current + a[i]);
        best = max(best, current);
    }

    cout << best;
    return 0;
}
```