Week 1: Basics & Implementation

Topics: - Input/Output, Loops, Conditionals - Arrays, Strings, Basic Math - Simple sorting

Weekly Tips: - Focus on writing clean, readable code. - Always test edge cases (0, 1, negative numbers, large numbers). - Use online judge IDE or local compiler to verify behavior.

Week 2: Ad-hoc & Simulation

Topics: - Simulation - Ad-hoc logic problems - Greedy basics

Weekly Tips: - Think step by step, simulate processes on paper first. - Carefully read problem constraints to optimize loops. - Greedy approach works if problem guarantees local optimality leads to global optimality.

Week 3: Sorting & Searching

Topics: - Sorting algorithms: QuickSort, MergeSort, STL sort - Binary Search & Ternary Search - Two-pointer technique

Weekly Tips: - Always check if STL sort suffices before implementing manually. - Binary search can be applied to sorted arrays or answer space. - Two-pointer technique is useful for finding pairs, sums, or sliding windows.

Problem 1: Forming Quiz Teams

Link: UVa 10911 **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>
using namespace std;

struct Student { double x, y; };

double distance(Student a, Student b) {
    return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}

int main() {
```

```
    int n;
    while (cin >> n && n != 0) {
        vector<Student> students(n);
        for (int i = 0; i < n; i++) cin >> students[i].x >> students[i].y;
        vector<int> idx(n);
        for (int i = 0; i < n; i++) idx[i] = i;
        double minSum = 1e9;
        do {
            double sum = 0;
            for (int i = 0; i < n; i+=2) {
                sum += distance(students[idx[i]], students[idx[i+1]]);
            }
            minSum = min(minSum, sum);
        } while (next_permutation(idx.begin(), idx.end()));
        printf("%.2f\n", minSum);
    }
    return 0;
}
```

Explanation Comments: - Generates all permutations to pair students; calculates sum of distances. - `next_permutation` ensures all pairings are checked. - Uses `sqrt` to calculate Euclidean distance. - Demonstrates brute-force combinatorial approach with sorting.

Problem 2: Age Sort

Link: UVa 11462 **Difficulty:** Beginner

C++ Solution with Explanation Comments:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    int n;
    while (cin >> n && n != 0) {
        vector<int> ages(n);
        for (int i = 0; i < n; i++) cin >> ages[i];
        sort(ages.begin(), ages.end()); // Sort using STL
        for (int i = 0; i < n; i++) {
            if (i > 0) cout << " ";
            cout << ages[i];
        }
        cout << endl;
```

```
    }
    return 0;
}
```

Explanation Comments: - STL `sort` handles sorting efficiently. - Handles multiple test cases. - Demonstrates basic array sorting and formatted output.

Problem 3: Aggressive Cows

Link: SPOJ AGGRCOW **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

bool canPlace(const vector<int>& stalls, int cows, int minDist) {
    int count = 1, lastPos = stalls[0];
    for (int i = 1; i < stalls.size(); i++) {
        if (stalls[i] - lastPos >= minDist) {
            count++;
            lastPos = stalls[i];
        }
    }
    return count >= cows;
}

int main() {
    int t; cin >> t;
    while (t--) {
        int n, c; cin >> n >> c;
        vector<int> stalls(n);
        for (int i = 0; i < n; i++) cin >> stalls[i];
        sort(stalls.begin(), stalls.end());
        int left = 1, right = stalls[n-1] - stalls[0], ans = 0;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (canPlace(stalls, c, mid)) {
                ans = mid;
                left = mid + 1;
            } else {
                right = mid - 1;
            }
```

```
        }
        cout << ans << endl;
    }
    return 0;
}
```

Explanation Comments: - Uses **binary search on answer space** to find max minimum distance. - `canPlace` checks if cows can be placed with given distance. - Classic example of applying binary search beyond simple arrays.

End of Week 3 - Focus on mastering sorting algorithms and binary search. - Try practicing two-pointer problems with sum/array constraints. - Understand difference between searching in array vs searching in answer space.