### 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.

### Problem 1: Exact Sum

**Link:** [UVa 11057](https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1998) **Difficulty:** Beginner/Intermediate

**C++ Solution with Explanation Comments:**

#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
  
int main() {  
 int n;  
 while (cin >> n) {  
 vector<int> coins(n);  
 for (int i = 0; i < n; i++) cin >> coins[i];  
 int m; cin >> m;  
 sort(coins.begin(), coins.end()); // Sort to use two-pointer technique  
  
 int left = 0, right = n-1;  
 int bestSum = 0, bestA = 0, bestB = 0;  
  
 // Two-pointer approach to find pair sum closest to m  
 while (left < right) {  
 int sum = coins[left] + coins[right];  
 if (sum > m) {  
 right--;  
 } else {  
 if (sum > bestSum) {  
 bestSum = sum;  
 bestA = coins[left];  
 bestB = coins[right];  
 }  
 left++;  
 }  
 }  
 cout << "Peter should buy books whose prices are " << bestA << " and " << bestB << ".\n\n";  
 }  
 return 0;  
}

**Explanation Comments:** - Sort array to efficiently find pair with sum <= m. - Two-pointer method avoids O(n^2) brute-force. - Keep track of best sum and corresponding pair.

### Problem 2: List of Conquests

**Link:** [UVa 10420](https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1393) **Difficulty:** Beginner

**C++ Solution with Explanation Comments:**

#include <iostream>  
#include <map>  
#include <string>  
using namespace std;  
  
int main() {  
 int n; cin >> n;  
 map<string,int> countryCount;  
 string country, name;  
  
 for (int i = 0; i < n; i++) {  
 cin >> country >> name;  
 countryCount[country]++; // Increment the number of people from each country  
 }  
  
 for (auto it = countryCount.begin(); it != countryCount.end(); it++) {  
 cout << it->first << " " << it->second << endl;  
 }  
  
 return 0;  
}

**Explanation Comments:** - Use map to automatically sort countries alphabetically. - Count occurrences while reading input. - Simple ad-hoc aggregation problem.

### Problem 3: Train Timetable

**Link:** [Kattis Train Timetable](https://open.kattis.com/problems/traintimetable) **Difficulty:** Intermediate

**C++ Solution with Explanation Comments:**

#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
  
struct Train { int start, end; };  
  
int main() {  
 int n; cin >> n;  
 vector<Train> trains(n);  
 for (int i = 0; i < n; i++) {  
 cin >> trains[i].start >> trains[i].end;  
 }  
  
 // Sort trains by end time to schedule as many as possible  
 sort(trains.begin(), trains.end(), [](Train a, Train b){ return a.end < b.end; });  
  
 int count = 0, lastEnd = 0;  
 for (auto t : trains) {  
 if (t.start >= lastEnd) { // Can take this train  
 lastEnd = t.end;  
 count++;  
 }  
 }  
 cout << count << endl;  
 return 0;  
}

**Explanation Comments:** - Greedy strategy: always pick the train that ends earliest. - Sorting by end time guarantees maximal number of non-overlapping trains. - Example of classic activity selection problem.

### Problem 4: Time Loop

**Link:** [Kattis Time Loop](https://open.kattis.com/problems/timeloop) **Difficulty:** Beginner

**C++ Solution with Explanation Comments:**

#include <iostream>  
using namespace std;  
  
int main() {  
 int n; cin >> n;  
 for (int i = 1; i <= n; i++) {  
 cout << i << " Abracadabra" << endl; // Repeat line number with fixed string  
 }  
 return 0;  
}

**Explanation Comments:** - Simple loop from 1 to n. - Concatenate loop counter with string. - Useful for practicing loops and formatting output.

**End of Week 2** - Practice simulation and greedy problems. - Test your understanding by writing explanations in comments yourself. - Try modifying problems to explore edge cases or alternative solutions.