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 9: Greedy & Interval Problems

Topics: - Activity Selection Problem - Interval Scheduling - Interval Covering - Fractional Knapsack

Weekly Tips: - Always sort intervals by finishing time for scheduling problems. - Greedy works when local optimum leads to global optimum. - Pay attention to edge cases where intervals overlap. - Fractional Knapsack can be solved using sorting by value/weight ratio.

Problem 1: Activity Selection Link: CSES Activities **Difficulty:** Beginner

C++ Solution with Explanation Comments:

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Activity{
    int start,end;
};
int main(){
    int n; cin>>n;
    vector<Activity> a(n);
    for(int i=0;i<n;i++) cin>>a[i].start>>a[i].end;
    sort(a.begin(),a.end(),[](Activity x,Activity y){return x.end<y.end;});
    int count=0,lastEnd=0;
    for(auto act:a){
        if(act.start>=lastEnd){
            count++;
            lastEnd=act.end;
        }
    }
    cout<<count<<endl;
}
```

Explanation Comments: - Sort activities by ending time to select the earliest finishing ones. - Iterate through and pick activities that start after the last selected ends. - Greedy ensures the maximum number of non-overlapping activities.

Problem 2: Fractional Knapsack Link: Hackerearth Fractional Knapsack **Difficulty:** Intermediate

C++ Solution with Explanation Comments:

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Item{
    double value,weight;
};
int main(){
    int n; cin>>n;
    vector<Item> items(n);
    for(int i=0;i<n;i++) cin>>items[i].value>>items[i].weight;
    double W; cin>>W;
    sort(items.begin(),items.end(),[](Item a,Item b){return a.value/a.weight >
b.value/b.weight;});
    double total=0;
    for(auto it:items){
        if(W>=it.weight){
            total+=it.value;
            W-=it.weight;
        }else{
            total+=it.value*(W/it.weight);
            break;
        }
    }
    cout<<fixed<<setprecision(2)<<total<<endl;
}
```

Explanation Comments: - Sort items by value-to-weight ratio in descending order. - Pick as much of the item as possible; take fractions if needed. - Greedy ensures maximum total value for fractional knapsack.

End of Week 9 - Focus on interval-based greedy problems. - Understand the conditions under which greedy algorithms produce optimal results. - Practice both scheduling and fractional selection problems.