import java.util.\*;

import java.io.\*;

public class BerkeleyAlgorithm {

static class ClockSynchronizer extends Thread {

private int nodeId;

private int clock;

private List<Integer> clockList;

private int masterNode;

private int adjustedTime;

public ClockSynchronizer(int nodeId, int clock, List<Integer> clockList, int masterNode) {

this.nodeId = nodeId;

this.clock = clock;

this.clockList = clockList;

this.masterNode = masterNode;

this.adjustedTime = clock; // initially adjusted time is same as current time

}

@Override

public void run() {

try {

// If the node is the master node, calculate the average time

if (masterNode == nodeId) {

System.out.println("Master Node " + nodeId + " is synchronizing clocks...");

// Master collects time from all nodes (including itself)

int sum = 0;

for (int i = 0; i < clockList.size(); i++) {

sum += clockList.get(i);

System.out.println("Node " + i + " Time: " + clockList.get(i));

}

// Calculate average time

int averageTime = sum / clockList.size();

System.out.println("Average Time: " + averageTime);

// Adjust the clocks of all nodes based on average time

for (int i = 0; i < clockList.size(); i++) {

int adjustment = averageTime - clockList.get(i);

clockList.set(i, clockList.get(i) + adjustment);

System.out.println("Adjusting Node " + i + " Time to: " + clockList.get(i));

}

} else {

// Non-master nodes wait for synchronization request and adjust clock based on master's avg time

// For simplicity, here, we assume all nodes get synchronized at same time

System.out.println("Node " + nodeId + " Initial Time: " + clock);

adjustedTime = clock + (clockList.get(masterNode) - clock); // Adjust time based on master's clock

System.out.println("Node " + nodeId + " Adjusted Time: " + adjustedTime);

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Step 1: Input the number of nodes

System.out.print("Enter the number of nodes: ");

int numNodes = sc.nextInt();

List<Integer> clockList = new ArrayList<>();

// Step 2: Input the initial clock values for each node

for (int i = 0; i < numNodes; i++) {

System.out.print("Enter time for Node " + i + ": ");

int nodeTime = sc.nextInt();

clockList.add(nodeTime);

}

// Step 3: Input the master node index

System.out.print("Enter the master node index: ");

int masterNode = sc.nextInt();

// Step 4: Start synchronization process

for (int i = 0; i < clockList.size(); i++) {

ClockSynchronizer synchronizer = new ClockSynchronizer(i, clockList.get(i), clockList, masterNode);

synchronizer.start();

}

sc.close();

}

}