#include <stdio.h>

#include <stdint.h>

#include <stdlib.h>

#define NUM\_TRANSFORMERS 3 // Number of transformers

#define MAX\_LOAD 100.0f // Maximum load capacity for each transformer

// Structure to hold transformer data

typedef struct {

float currentLoad; // Current load of the transformer (in amperes or any unit)

float maxLoad; // Maximum load of the transformer

int relayStatus; // Relay control: 0 - off, 1 - on

} Transformer;

// Array to store data for each transformer

Transformer transformers[NUM\_TRANSFORMERS];

// Function prototypes

void measureLoad();

void redistributeLoad();

void controlRelay(int transformerIndex, int status);

void printTransformerStatus();

int main() {

// Initialize transformer data

for (int i = 0; i < NUM\_TRANSFORMERS; i++) {

transformers[i].currentLoad = 0.0f;

transformers[i].maxLoad = MAX\_LOAD;

transformers[i].relayStatus = 0; // All transformers are initially off

}

while (1) {

// Step 1: Measure load

measureLoad();

// Step 2: Check if any transformer is overloaded and redistribute load

redistributeLoad();

// Step 3: Print status of transformers

printTransformerStatus();

// Small delay to simulate real-time monitoring

\_delay\_ms(1000);

}

return 0;

}

// Function to measure the load (simulated for this example)

void measureLoad() {

// Simulate load measurement by random values (replace with actual sensor readings)

for (int i = 0; i < NUM\_TRANSFORMERS; i++) {

transformers[i].currentLoad = rand() % 120; // Random load between 0-120 units

}

}

// Function to redistribute load across transformers

void redistributeLoad() {

for (int i = 0; i < NUM\_TRANSFORMERS; i++) {

if (transformers[i].currentLoad > transformers[i].maxLoad) {

printf("Transformer %d overloaded, redistributing load.\n", i);

// Find the transformer with the least load to transfer the excess load

for (int j = 0; j < NUM\_TRANSFORMERS; j++) {

if (i != j && transformers[j].currentLoad < transformers[j].maxLoad) {

// Calculate how much load to transfer