#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> // for sleep function

#define N 5 // Number of philosophers

#define ITERATIONS 10 // Number of iterations to run the simulation

enum {THINKING, HUNGRY, EATING} state[N];

int philosophers[N] = {0, 1, 2, 3, 4};

// Function declarations

void test(int);

void pickup\_forks(int);

void return\_forks(int);

void think(int);

void eat(int);

void simulate\_dining(int);

int main() {

// Initialize the state of each philosopher

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

state[i] = THINKING;

}

simulate\_dining(ITERATIONS);

return 0;

}

// Function to simulate dining philosophers

void simulate\_dining(int iterations) {

int i = 0;

for (int j = 0; j < iterations; j++) {

think(philosophers[i]);

pickup\_forks(philosophers[i]);

eat(philosophers[i]);

return\_forks(philosophers[i]);

i = (i + 1) % N; // Move to the next philosopher

}

}

// Function for philosophers to think

void think(int philosopher) {

printf("Philosopher %d is thinking.\n", philosopher);

sleep(1); // Simulate thinking

}

// Function for philosophers to eat

void eat(int philosopher) {

printf("Philosopher %d is eating.\n", philosopher);

sleep(1); // Simulate eating

}

// Function to pick up forks

void pickup\_forks(int philosopher) {

state[philosopher] = HUNGRY;

printf("Philosopher %d is hungry.\n", philosopher);

test(philosopher);

while (state[philosopher] != EATING) {

// Busy wait

}

}

// Function to return forks

void return\_forks(int philosopher) {

state[philosopher] = THINKING;

printf("Philosopher %d is putting down forks.\n", philosopher);

test((philosopher + N - 1) % N); // Test left neighbor

test((philosopher + 1) % N); // Test right neighbor

}

// Function to test if a philosopher can eat

void test(int philosopher) {

if (state[philosopher] == HUNGRY &&

state[(philosopher + N - 1) % N] != EATING &&

state[(philosopher + 1) % N] != EATING) {

state[philosopher] = EATING;

printf("Philosopher %d takes forks %d and %d.\n", philosopher, philosopher, (philosopher + 1) % N);

}

}

