**LAB-5**

**Dining-Philosophers problem**

1. Write a C program to simulate the concept of Dining-Philosophers

problem.

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

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

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (i + 4) % N

#define RIGHT (i + 1) % N

int state[N];

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

sem\_t mutex;

sem\_t S[N];

void test(int i) {

if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {

state[i] = EATING;

sleep(2);

printf("Philosopher %d takes fork %d and %d\n", i + 1, LEFT + 1, i + 1);

printf("Philosopher %d is Eating\n", i + 1);

sem\_post(&S[i]);

}

}

void take\_fork(int i) {

sem\_wait(&mutex);

state[i] = HUNGRY;

printf("Philosopher %d is Hungry\n", i + 1);

test(i);

sem\_post(&mutex);

sem\_wait(&S[i]);

sleep(1);

}

void put\_fork(int i) {

sem\_wait(&mutex);

state[i] = THINKING;

printf("Philosopher %d putting fork %d and %d down\n", i + 1, LEFT + 1, i + 1);

printf("Philosopher %d is thinking\n", i + 1);

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void\* philosopher(void\* num) {

while (1) {

int\* i = num;

sleep(1);

take\_fork(\*i);

sleep(0);

put\_fork(\*i);

}

}

int main() {

int i;

pthread\_t thread\_id[N];

sem\_init(&mutex, 0, 1);

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

sem\_init(&S[i], 0, 0);

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

pthread\_create(&thread\_id[i], NULL, philosopher, &phil[i]);

printf("Philosopher %d is thinking\n", i + 1);

}

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

pthread\_join(thread\_id[i], NULL);

}

}

OUTPUT:

