6. [Process Synchronization]

Considered there are N philosophers seated around a circular table CO3 with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both. Write a program to solve the problem using process synchronization technique

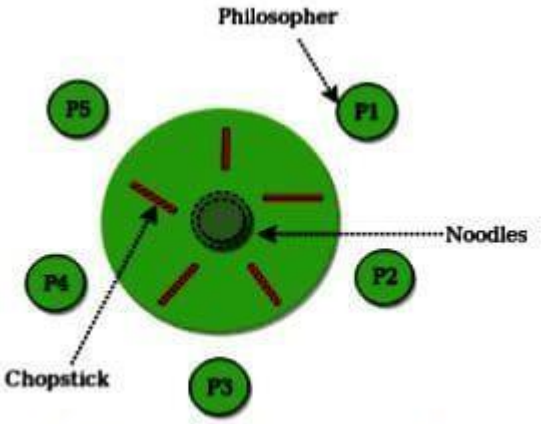

Fig: 1. Dining Philosopher Problem

CODE:

```
 GNU nano 8.7                                  chopstick.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5

sem_t chopstick[N];
pthread_t philosopher[N];

void* eat(void* arg) {
    int id = *(int*)arg;

    printf("Philosopher %d is thinking\n", id);
    sleep(1);

    sem_wait(&chopstick[id]);
    sem_wait(&chopstick[(id + 1) % N]);

    printf("Philosopher %d is eating\n", id);
    sleep(1);

    sem_post(&chopstick[id]);
    sem_post(&chopstick[(id + 1) % N]);

    printf("Philosopher %d finished eating\n", id);
    return NULL;
}

int main() {
    int i, id[N];

    for (i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);

    for (i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&philosopher[i], NULL, eat, &id[i]);
    }

    for (i = 0; i < N; i++)
        pthread_join(philosopher[i], NULL);

    return 0;

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket  M-B Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was    M-F Next
```

```
GNU nano 8.7                                chopstick.c
#include <unistd.h>

#define N 5

sem_t chopstick[N];
pthread_t philosopher[N];

void* eat(void* arg) {
    int id = *(int*)arg;

    printf("Philosopher %d is thinking\n", id);
    sleep(1);

    sem_wait(&chopstick[id]);
    sem_wait(&chopstick[(id + 1) % N]);

    printf("Philosopher %d is eating\n", id);
    sleep(1);

    sem_post(&chopstick[id]);
    sem_post(&chopstick[(id + 1) % N]);

    printf("Philosopher %d finished eating\n", id);
    return NULL;
}

int main() {
    int i, id[N];

    for (i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);

    for (i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&philosopher[i], NULL, eat, &id[i]);
    }

    for (i = 0; i < N; i++)
        pthread_join(philosopher[i], NULL);

    return 0;
}


^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket  M-B Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^B Where Was    M-F Next
```

OUTPUT :

```
Sakshi Wadiyalwar@DESKTOP-JJN9O8K MINGW64 ~
$ gcc --version
gcc (GCC) 15.2.0
Copyright (C) 2025 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Sakshi Wadiyalwar@DESKTOP-JJN9O8K MINGW64 ~
$ nano dining.c

Sakshi Wadiyalwar@DESKTOP-JJN9O8K MINGW64 ~
$ gcc dining.c -o dining -pthread

Sakshi Wadiyalwar@DESKTOP-JJN9O8K MINGW64 ~
$ ./dining
Philosopher 0 is Thinking
Philosopher 1 is Thinking
Philosopher 2 is Thinking
Philosopher 3 is Thinking
Philosopher 4 is Thinking
Philosopher 2 is Hungry
Philosopher 2 takes chopsticks 1 and 2
Philosopher 2 is Eating
Philosopher 0 is Hungry
Philosopher 0 takes chopsticks 4 and 0
Philosopher 0 is Eating
Philosopher 1 is Hungry
Philosopher 4 is Hungry
Philosopher 3 is Hungry
Philosopher 2 putting chopsticks 1 and 2 down
Philosopher 2 is Thinking
Philosopher 3 takes chopsticks 2 and 3
Philosopher 3 is Eating
Philosopher 0 putting chopsticks 4 and 0 down
Philosopher 0 is Thinking
Philosopher 1 takes chopsticks 0 and 1
Philosopher 1 is Eating
Philosopher 2 is Hungry
Philosopher 0 is Hungry
Philosopher 3 putting chopsticks 2 and 3 down
Philosopher 3 is Thinking
Philosopher 4 takes chopsticks 3 and 4
Philosopher 4 is Eating
Philosopher 1 putting chopsticks 0 and 1 down
Philosopher 1 is Thinking
Philosopher 2 takes chopsticks 1 and 2
Philosopher 2 is Eating
Philosopher 3 is Hungry
```