WEEK 6

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

CODE :

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>

#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N

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

sem_t mutex;
sem_t S[N];

void test(int phnum)
{
    if (state[phnum] == HUNGRY
```

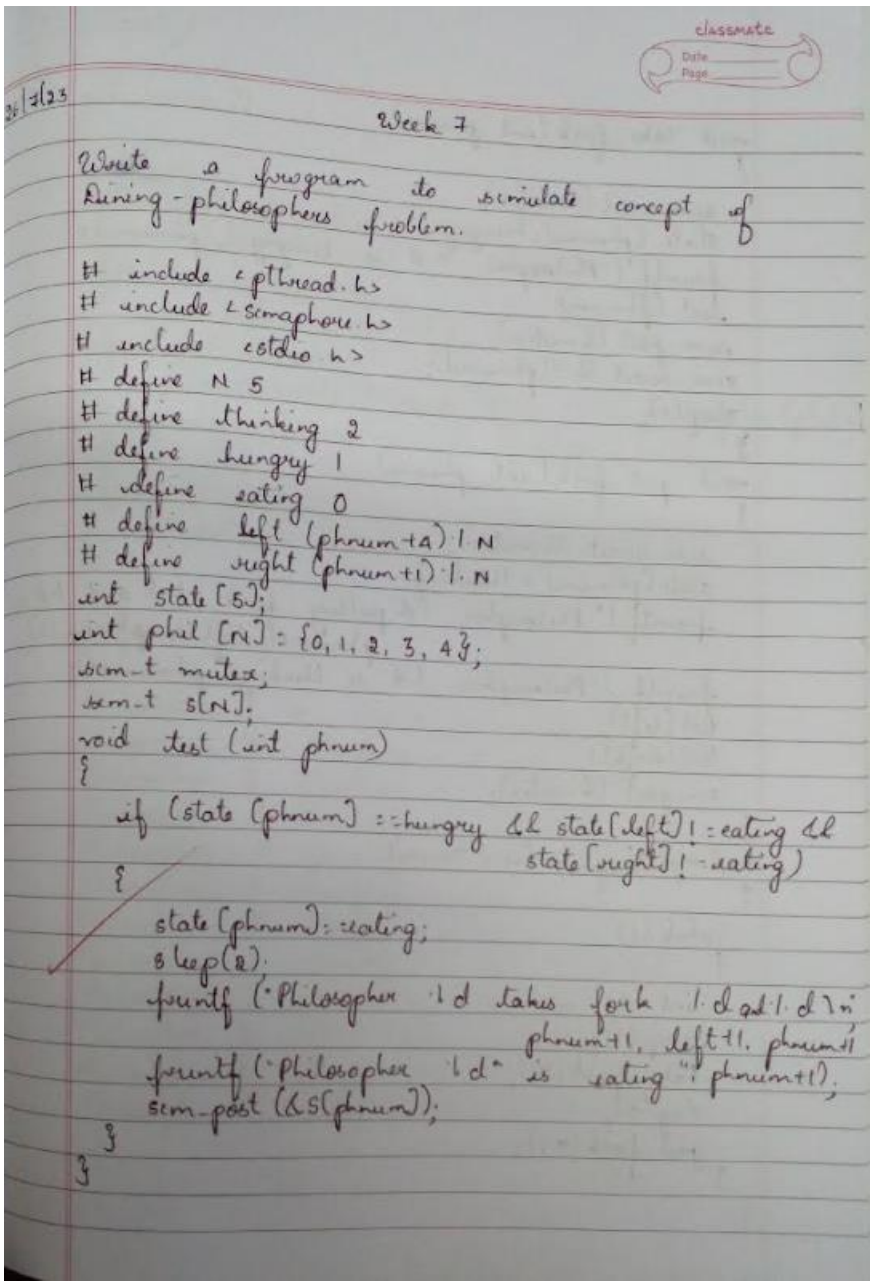
```
        && state[LEFT] != EATING
        && state[RIGHT] != EATING) {

        state[phnum] = EATING;

        sleep(2);

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

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


        sem_post(&S[phnum]);
    }
}


void take_fork(int phnum)
{

    sem_wait(&mutex);


    state[phnum] = HUNGRY;
```

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

    test(phnum);

    sem_post(&mutex);


    sem_wait(&S[phnum]);

    sleep(1);
}


void put_fork(int phnum)
{

    sem_wait(&mutex);


    state[phnum] = THINKING;

    printf("Philosopher %d putting fork %d and %d down\n",
          phnum + 1, LEFT + 1, phnum + 1);
    printf("Philosopher %d is thinking\n", phnum + 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);
}
```

OBSERVATION :



Week 7

Write a program to simulate concept of Dining-philosophers problem.

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#define N 5
#define thinking 2
#define hungry 1
#define eating 0
#define left (phnum+4)%N
#define right (phnum+1)%N
int state[5];
int phil[N] = {0,1,2,3,4};
sem_t mutex;
sem_t s[N];
void test (int phnum)
{
    if (state[phnum] == hungry && state[left] != eating &&
                                  state[right] != eating)
    {
        state[phnum] = eating;
        sleep(2);
        printf("Philosopher %d takes fork %d and %d\n",
                                  phnum+1, left+1, phnum+1);
        printf("Philosopher %d is eating\n", phnum+1);
        sem_post(&s[phnum]);
    }
}
```

```
void take_fork (int phnum)
{
  sem_wait (&mutex);
  state [phnum] = hungry;
  printf ("Philosopher %d is hungry", phnum+1);
  test (phnum);
  sem_post (&mutex);
  sem_wait (&S[phnum]);
  sleep(1);
}
void put_fork (int phnum)
{
  sem_wait (&mutex);
  state [phnum] = thinking;
  printf ("Philosopher %d putting fork %d and %d d
                    phnum+1, left+1, phnum+1);
  printf ("Philosopher %d is thinking", phnum+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
      printf ("Philosopher %d thinking", i+1);
  }
  for (i=0; i<N; i++)
      pthread_join (thread_id [i], NULL);
}
```

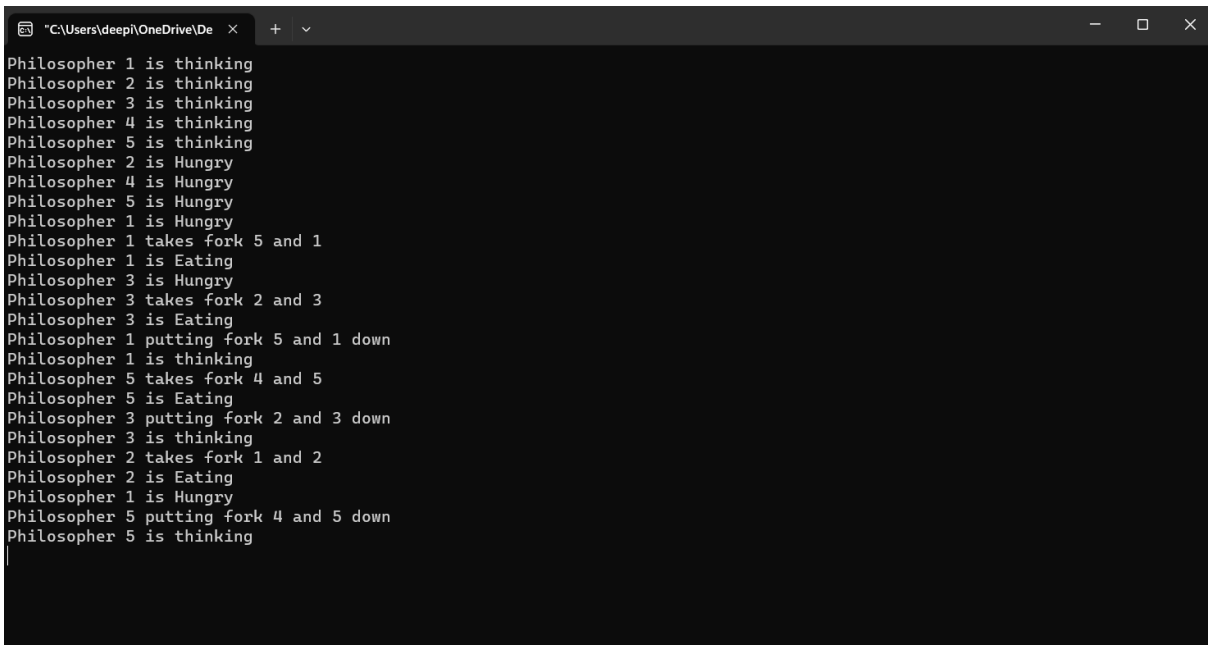
Output :

```
Philosopher 1 is thinking
Philosopher 2 thinking
Philosopher 3 thinking
Philosopher 4 thinking
Philosopher 5 thinking
Philosopher 1 hungry
Philosopher 2 hungry
Philosopher 3 hungry
Philosopher 5 hungry
Philosopher 5 take fork 4 and 5
Philosopher 5 eating
Philosopher 3 hungry
```

10/10

N
2/10/23

OUTPUT :


"C:\Users\deepi\OneDrive\De
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 2 is Hungry
Philosopher 4 is Hungry
Philosopher 5 is Hungry
Philosopher 1 is Hungry
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 3 is Hungry
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 1 is Hungry
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking