TE IT

Name   : Omkar Gurav

Roll no : 8048

Deadlock Avoidance Using Semaphores: Implement the deadlock-free solution to Dining Philosophers problem to illustrate the problem of deadlock and/or starvation that can occur when many synchronized threads are competing for limited resources.

```
#include<stdio.h>

#include<semaphore.h>

#include<pthread.h>


#define N 5

#define THINKING 0

#define HUNGRY 1

#define EATING 2

#define LEFT (ph_num+4)%N

#define RIGHT (ph_num+1)%N


sem_t mutex;

sem_t S[N];


int state[N];

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


void *philospher(void *num)
```

```
{
    while(1)
    {
        int *i = num;
        sleep(1);
        take_fork(*i);
        sleep(0);
        put_fork(*i);
    }
}

void take_fork(int ph_num)
{
    sem_wait(&mutex);
    state[ph_num] = HUNGRY;
    printf("Philosopher %d is Hungry \n",ph_num+1);
    test(ph_num);
    sem_post(&mutex);
    sem_wait(&S[ph_num]);
    sleep(1);
}

void test(int ph_num)
{
    if(state[ph_num] == HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING)
    {
        state[ph_num] = EATING;
        sleep(2);
        printf("Philosopher %d Takes Fork %d and %d
\n",ph_num+1,LEFT+1,ph_num+1);
        printf("Philosopher %d is Eating \n",ph_num+1);
        sem_post(&S[ph_num]);
    }
}

void put_fork(int ph_num)
{
    sem_wait(&mutex);
    state[ph_num] = THINKING;
    printf("Philosopher %d Putting Fork %d and %d Down
\n",ph_num+1,LEFT+1,ph_num+1);
    printf("Philosopher %d is Thinking \n",ph_num+1);
    test(LEFT);
    test(RIGHT);
    sem_post(&mutex);
}

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,philospher,&phil_num[i]);
        printf("Philosopher %d is Thinking \n",i+1);
    }
    for(i=0;i<N;i++)
    {
        pthread_join(thread_id[i],NULL);
    }
}
```

Output:

```
Philosopher 1 is Thinking
Philosopher 2 is Thinking
Philosopher 3 is Thinking
Philosopher 4 is Thinking
Philosopher 5 is Thinking
Philosopher 1 is Hungry
Philosopher 1 Takes Fork 5 and 1
Philosopher 1 is Eating
Philosopher 2 is Hungry
Philosopher 4 is Hungry

Philosopher 4 Takes Fork 3 and 4

Philosopher 4 is Eating

Philosopher 3 is Hungry

Philosopher 5 is Hungry

Philosopher 1 Putting Fork 5 and 1 Down

Philosopher 1 is Thinking

Philosopher 2 Takes Fork 1 and 2

Philosopher 2 is Eating

Philosopher 4 Putting Fork 3 and 4 Down

Philosopher 4 is Thinking

Philosopher 5 Takes Fork 4 and 5

Philosopher 5 is Eating

Philosopher 1 is Hungry

Philosopher 2 Putting Fork 1 and 2 Down

Philosopher 2 is Thinking

Philosopher 3 Takes Fork 2 and 3

Philosopher 3 is Eating

Philosopher 4 is Hungry

Philosopher 5 Putting Fork 4 and 5 Down

Philosopher 5 is Thinking

Philosopher 1 Takes Fork 5 and 1

Philosopher 1 is Eating

Philosopher 3 Putting Fork 2 and 3 Down

Philosopher 3 is Thinking

Philosopher 4 Takes Fork 3 and 4

Philosopher 4 is Eating
Philosopher 2 is Hungry
Philosopher 5 is Hungry
Philosopher 1 Putting Fork 5 and 1 Down
Philosopher 1 is Thinking
Philosopher 2 Takes Fork 1 and 2
Philosopher 2 is Eating
Philosopher 3 is Hungry
Philosopher 4 Putting Fork 3 and 4 Down
Philosopher 4 is Thinking
Philosopher 5 Takes Fork 4 and 5
Philosopher 5 is Eating
Philosopher 2 Putting Fork 1 and 2 Down
Philosopher 2 is Thinking
Philosopher 3 Takes Fork 2 and 3
Philosopher 3 is Eating
Philosopher 1 is Hungry
Philosopher 4 is Hungry
Philosopher 5 Putting Fork 4 and 5 Down
Philosopher 5 is Thinking
Philosopher 1 Takes Fork 5 and 1
Philosopher 1 is Eating
Philosopher 2 is Hungry
Philosopher 3 Putting Fork 2 and 3 Down
Philosopher 3 is Thinking
Philosopher 4 Takes Fork 3 and 4
Philosopher 4 is Eating
Philosopher 5 is Hungry
Philosopher 1 Putting Fork 5 and 1 Down

Philosopher 1 is Thinking

Philosopher 2 Takes Fork 1 and 2

Philosopher 2 is Eating

Philosopher 3 is Hungry

Philosopher 4 Putting Fork 3 and 4 Down

Philosopher 4 is Thinking
```