```
#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];
 void * philospher(void *num);   void take_fork(int);
void put_fork(int);        void test(int);
int state[N];
int phil_num[N]={0,1,2,3,4};
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);
}
```

```
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);
}
```

```
#include<stdio.h>        #include<pthread.h>
#include<semaphore.h>           sem_t  mutex, writeblock;
int data = 0,rcount = 0;
int main()
{   int i,b;
  pthread_t  rtid[5],wtid[5];
  sem_init(&mutex,0,1);     sem_init(&writeblock,0,1);
  for(i=0;i<=2;i++)
  {    pthread_create(&wtid[i],NULL,writer,(void *)i);
    pthread_create(&rtid[i],NULL,reader,(void *)i);
  }
  for(i=0;i<=2;i++)
  {pthread_join(wtid[i],NULL);
    pthread_join(rtid[i],NULL);
  }   return 0; }
```

```
void *reader(void *arg)
{   int f;
  f = ((int)arg);
  sem_wait(&mutex);
rcount = rcount + 1;
  if(rcount==1)
 sem_wait(&writeblock);
  sem_post(&mutex);
  printf("Data read by the reader%d is %d\n",f,data);
  sleep(1);
  sem_wait(&mutex);
  rcount = rcount - 1;
  if(rcount==0)
   sem_post(&writeblock);
  sem_post(&mutex);
}
```

```
void *writer(void *arg)
{
  int f;
  f = ((int) arg);
  sem_wait(&writeblock);
  data++;
  printf("Data writen by the writer%d is %d\n",f,data);
  sleep(1);
  sem_post(&writeblock);
}
```