Int main()4. Apply the concepts of Process/Thread synchronization to build Applications to demonstrate process/thread synchronization using semaphores and mutex. Implement Dining philosophers problem.

#include<stdio.h> #include<stdlib.h> #include<pthread.h> #include<semaphore.h> #include<unistd.h> sem\_t chopstick[5]; void \* philos(void \*); void eat(int);

int main()

{

int i,n[5]; pthread\_t T[5]; for(i=0;i<5;i++)

sem\_init(&chopstick[i],0,1); for(i=0;i<5;i++){

n[i]=i;

pthread\_create(&T[i],NULL,philos,(void \*)&n[i]);

}

for(i=0;i<5;i++)

pthread\_join(T[i],NULL);

}

void \* philos(void \* n)

{

int ph=\*(int \*)n;

printf("Philosopher %d wants to eat\n",ph); printf("Philosopher %d tries to pick left chopstick\n",ph); sem\_wait(&chopstick[ph]);

printf("Philosopher %d picks the left chopstick\n",ph); printf("Philosopher %d tries to pick the right chopstick\n",ph); sem\_wait(&chopstick[(ph+1)%5]);

printf("Philosopher %d picks the right chopstick\n",ph); eat(ph);

sleep(2);

printf("Philosopher %d has finished eating\n",ph); sem\_post(&chopstick[(ph+1)%5]);

printf("Philosopher %d leaves the right chopstick\n",ph); sem\_post(&chopstick[ph]);

printf("Philosopher %d leaves the left chopstick\n",ph);

}

void eat(int ph)

{

printf("Philosopher %d begins to eat\n",ph);

}