PAGE NO :
DATE :

#include write a c program to solve philoso-phers problem

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

sem_t *forks;
sem_t mutex;

void* philosopher_one(void* num){
    int id = *(int*) num;
    printf("P%d is waiting\n", id);
    sem_wait(&mutex);
    printf("P%d is granted to eat\n", id);
    sem_wait(&forks[id]);
    sem_wait(&forks[(id+1)%5]);
    sem_post(&mutex);
    sleep(1);

    sem_post(&forks[id]);
    sem_post(&forks[(id+1)%5]);
    return NULL;
}
void* philosopher_two(void* num){
    int id = *(int*)num;
    printf("P%d is waiting\n", id);
    sem_wait(&forks[id]);
    sem_wait(&forks[(id+1)%5]);
    printf("P%d is granted to eat\n", id);
    sleep(1);
```

```
    sem_post(&forks[id]);
    sem_post(&forks[(id+1)%5]);
    return NULL;
}
int main(){
    int num_philosophers, num_hungry;
    printf("Dining philosophers problem\n");
    printf("Enter the total no.of philosophers: ");
    scanf("%d", &numphilosophers);
    int hungry_positions[num_hungry];
    for(int i=0; i<num_hungry; i++){
        printf("Enter philosopher %d position
                i+1);
        scanf("%d", &num_hungry);
        hungry_positions[i]--;
    }
    forks = (sem_t*)malloc(num_philosophers
            sizeof(sem_t));
    pthread_t philosophers[num_hungry];
    int ids[num_hungry];

    for(int i=0; i<num_philosophers; i++){
        sem_init(&forks[i], 0, 1);
    }
    sem_init(&muter, 0, 1);
    int choice;
    printf("\n1. One can eat at a time \t 2. Two can eat at a time \t 3. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    if(choice==1){
        printf("\nAllow one philosopher to
                at any time");
```

```
for(int i=0; i<num_hungry; i++){
    ids[i]=hungry_position[i];
    pthread_create(&philosophers[i], NULL,
        Philosopher_one, &ids[i]);
  }
} else if(choice==2){
  printf("\n Allow two philosophers at
    a time \n");
  for(int i=0; i<num_hungry; i++){
    ids[i]=hungry_positions[i];
    pthread_create(&philosophers[i], NULL,
        philosopher_two, &ids[i]);
  }
} else {
  printf("Exiting program.\n");
  free(forks);
  return 0;
}
for(int i=0; i<num_hungry; i++){
  pthread_join(philosophers[i], NULL);
}
for(int i=0; i<num_philosophers; i++){
  sem_destroy(&forks[i]);
}
sem_destroy(&mutex);
free(forks);
return 0;
}
```

PAGE NO :
DATE :

Output

Dining philosopher problem.
Enter the total no. of philosophers: 5
How many are hungry: 5
Enter philosopher 1 position: 1
Enter philosopher 2 position: 2
Enter philosopher 3 position: 3
Enter philosopher 4 position: 4
Enter philosopher 5 position: 5

1. One can eat at a time 2. two can eat at a time 3. exit
Enter your choice: 1
Allow one philosophers to eat at any time.
P0 is waiting
P0 is granted to eat
P2 is waiting
P2 is granted to eat
P3 is waiting
P3 is granted to eat
P4 is waiting
P1 is waiting
P4 is granted to eat
P1 is granted to eat.