Assignment 11 | Operating System CE-092

Assignment submission for Operating System subject week 11.

nevilparmar24@gmail.com

Task 1:

Sleeping Barber Problem.

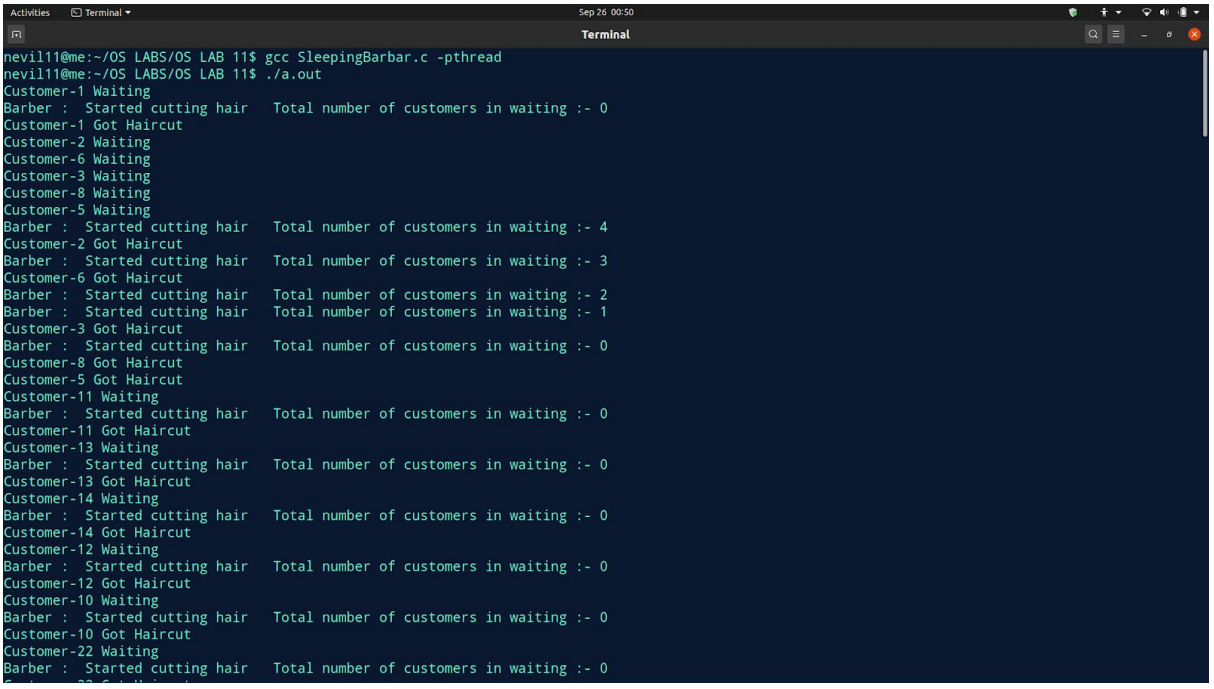
Code:

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

const int NO_CUSTOMERS = 100;
const int NO_BARBER = 1;
const int CAPACITY = 5;

sem_t sem_wait_count, sem_cust, sem_barb;
int waiting = 0;

void *Customer(void *arg)
{
    int * custId=(int *)arg;
    sem_wait(&sem_wait_count);
    if(waiting < CAPACITY)
    {
```

```
        waiting++;
        printf("Customer-%d Waiting \n", *custId);
        sem_post(&sem_cust);
        sem_post(&sem_wait_count);
        sem_wait(&sem_barb);
        printf("Customer-%d Got Haircut\n", *custId);
    }
    else
    {
        sem_post(&sem_wait_count);
    }
}


void *Barber(void *arg)
{
    while(1)
    {
        sem_wait(&sem_cust);
        sem_wait(&sem_wait_count);
        waiting--;
        printf("Barber :  Started cutting hair \t Total
number of customers in waiting :- %d \n", waiting);
        sem_post(&sem_barb);
        sem_post(&sem_wait_count);
    }
}

void main()
{
    int i, customerIds[NO_CUSTOMERS];
    pthread_t barberThreads[NO_BARBER],
customerThreads[NO_CUSTOMERS];
    sem_init(&sem_barb, 0, 0);
```

```
    sem_init(&sem_cust, 0, 0);
    sem_init(&sem_wait_count, 0, 1);

    for(i = 0 ; i < NO_BARBER ; i++)
    {
        pthread_create(&barberThreads[i], NULL,Barber,
(void*)NULL);
    }
    for(i = 0 ; i < NO_CUSTOMERS ; i++)
    {
        customerIds[i] = i + 1;
        pthread_create(&customerThreads[i],
NULL,Customer, (void*)&customerIds[i]);
    }
    for(i = 0 ; i < NO_BARBER ; i++)
    {
        pthread_join(barberThreads[i], NULL);
    }
    for(i = 0 ; i < NO_CUSTOMERS ; i++)
    {
        pthread_join(customerThreads[i], NULL);
    }
}
```

Output:



Task 2:

Dining Philosopher Problem.

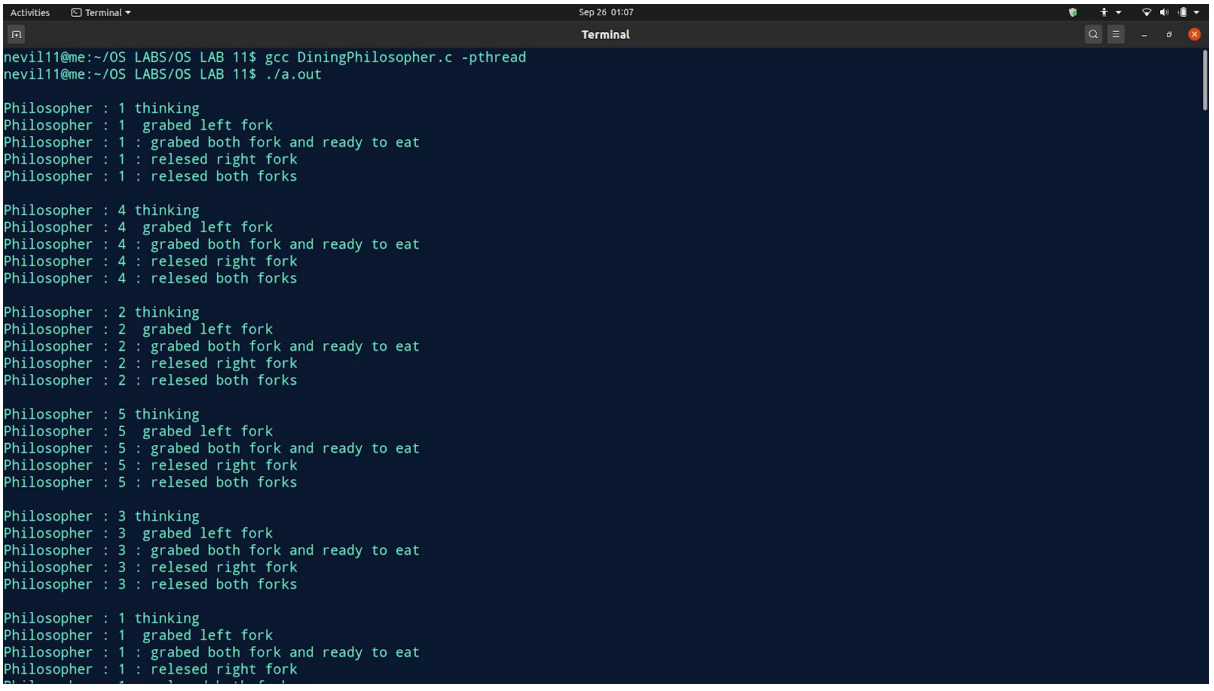
Code:

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

#define totalPhilosophers 5
sem_t sem_fork[totalPhilosophers], sem_room;

void *Philosopher(void *arg)
{
    int *ph_id = (int *)arg;
    while(1)
```

```
    {
        sem_wait(&sem_room);
        printf("\nPhilosopher : %d thinking\n",
*ph_id);
        sem_wait(&sem_fork[*ph_id - 1]);
        printf("Philosopher : %d  grabed left fork\n",
*ph_id);

sem_wait(&sem_fork[(*ph_id)%totalPhilosophers]);
        printf("Philosopher : %d : grabed both fork and
ready to eat\n", *ph_id);

sem_post(&sem_fork[(*ph_id)%totalPhilosophers]);
        printf("Philosopher : %d : relesed right
fork\n", *ph_id);
        sem_post(&sem_fork[*ph_id - 1]);
        printf("Philosopher : %d : relesed both
forks\n", *ph_id);
        sem_post(&sem_room);
        sleep(1);
    }
}

void main()
{
    pthread_t phi_thread[totalPhilosophers];
    int i, ph_id[totalPhilosophers];
    sem_init(&sem_room,0, totalPhilosophers-1);

    for(i = 0 ; i < totalPhilosophers ; i++)
    {
        sem_init(&sem_fork[i], 0, 1);
    }
    for(i = 0 ; i < totalPhilosophers ; i++)
```

```
    {
        ph_id[i] = i + 1;
        pthread_create(&phi_thread[i], NULL,
Philosopher, (void*)&ph_id[i]);
    }
    for(i = 0 ; i < totalPhilosophers ; i++)
    {
        pthread_join(phi_thread[i], NULL);
    }
}
```

Output:



Nevil Parmar
CE-092
https://nevilparmar.me