Student Name: Vishal Das
Student ID: 11702786
Email Address: vishal.11702786@lpu.in
GitHub Link: https://github.com/kindavishal/

Code: Mention solution code assigned to you
1. Explain the problem in terms of operating system concept? (Max 200 word)

Description:
- **The Sleeping Barber Problem:** The problem involves the use of locks and semaphores to avoid deadlock in the processes. In the solution, we make use of three semaphores. First is for the customer which counts the number of customers present in the waiting room (customer in the barber chair is not included because he is not waiting). Second, the barber 0 or 1 is used to tell whether the barber is idle or is working, And the third mutex is used to provide the mutual exclusion which is required for the process to execute. In the solution, the customer has the record of the number of customers waiting in the waiting room if the number of customers is equal to the number of chairs in the waiting room then the upcoming customer leaves the barbershop.
When the barber shows up in the morning, he executes the procedure barber, causing him to block on the semaphore customers because it is initially 0. Then the barber goes to sleep until the first customer comes up.

- **Priority Scheduling Algorithm:** Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.
Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

2. Write the algorithm for proposed solution of the assigned problem.

Algorithm:
- **Sleeping Barber Problem:** The barber waits for the customer to arrive, if customer doesn't arrive, he goes to sleep, which gives process to customer, releases lock. If customer count increases to 2, then process is switched to barber and barber cuts hair, and reduces customer count by 1 and puts lock. Next customer waits in the waiting room. After hair cut is done, barber releases lock and customer leaves. The process repeats until customer waiting is 0.

- **Priority Scheduling Algorithm:** In this, each process is assigned a priority and processes are executed on the basis of their priority. We can either choose to set priority of the lowest number to be the first priority or vice versa. No other process can execute until the process with the highest priority has fully executed. If two processes have same priority, then process is executed on the basis of their arrival time.

3. Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

Description (purpose of use):
- Complexity of Sleeping barber problem: **O(n)**
- Complexity of Priority Scheduling Algorithm: **O(n)**

4. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

Code snippet:
- **Sleeping barber problem** (written in C)**:**

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<sys/ipc.h>
#include<semaphore.h>
#define N 5
time_t end_time;
sem_t mutex,customers,barbers;
int count=0;
void barber(void *arg);
void customer(void *arg);
int main(int argc,char *argv[]) {
   pthread_t id1,id2;
   int status=0;
   end_time=time(NULL)+20;
   sem_init(&mutex,0,1);
   sem_init(&customers,0,1);
   sem_init(&barbers,0,1);
   pthread_create(&id2,NULL,(void *)customer,NULL);
   pthread_create(&id1,NULL,(void *)barber,NULL);
   pthread_join(id2,NULL);
   pthread_join(id1,NULL);
   exit(0);
}
void barber(void *arg) {
   while(time(NULL)<end_time || count>0) {
      sem_wait(&customers);
      sem_wait(&mutex);
      count--;
      printf("Barber is cutting hair, Customer count is: %d\n",count);
      sem_post(&mutex);
```

```
        sem_post(&barbers);
        sleep(3);
    }
}
void customer(void *arg) {
    while(time(NULL)<end_time) {
        sem_wait(&mutex);
        if(count<N) {
            count++;
            printf("More customer. Customer count is: %d\n",count);
            sem_post(&mutex);
            sem_post(&customers);
            sem_wait(&barbers);
        }
        else
            sem_post(&mutex);
        sleep(1);
    }
}
```

- Priority Scheduling Algorithm (written in Python3)**:**

```
def waitingTime(processes, n, wt):
    wt[0] = 0
    for i in range(1, n):
        wt[i] = processes[i - 1][1] + wt[i - 1]
def turnAroundTime(processes, n, wt, tat):
        for i in range(n):
        tat[i] = processes[i][1] + wt[i]
def findavgTime(processes, n):
    wt = [0] * n
    tat = [0] * n
    waitingTime(processes, n, wt)
    turnAroundTime(processes, n, wt, tat)
    print("\nProcesses Burst Time Waiting", "Time Turn-Around Time")
    total_wt = 0
```

```
    total_tat = 0
    for i in range(n):
        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print(" ", processes[i][0], "\t\t",
              processes[i][1], "\t\t",
              wt[i], "\t\t", tat[i])
    print("\nAverage waiting time = %.5f "%(total_wt /n))
    print("Average turn around time = ", total_tat / n)
def priorityScheduling(proc, n):
    proc = sorted(proc, key = lambda proc:proc[2],
                          reverse = True);
    print("Order in which processes gets executed")
    for i in proc:
        print(i[0], end = " ")
    findavgTime(proc, n)
    if __name__ =="__main__":
        process = [[1, 10, 1],
            [2, 20, 0],
            [3, 15, 1],
            [4, 11, 2]]
    n = 4
    priorityScheduling(process, n)
```

5. If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.
Description: No additional algorithm used.

6. Explain the boundary conditions of the implemented code.
Description:

- Constraints for Sleeping Barber Problem:

1. Only one customer can have haircut at a time
2. If there is no customer, then barber goes to sleep
3. If barber is already cutting hair of one customer, then other customers wait in the waiting room
4. If there is chair free in the waiting room, then the customer waits, otherwise leaves the shop without haircut.

- Constraints for Priority Scheduling Algorithm:

1. Process with highest priority gets preference first.
2. If two or more processes have same priority, then process if executed on their arrival time.

3. If there is already a process running and another process comes with higher priority, then the running process is pre-empted.

7. Explain all the test cases applied on the solution of assigned problem.

Description:
- Sleeping barber problem:

1. No customer -> Barber Sleeps:
   - If there is no customer in the waiting room, then barber will go to sleep.
2. Customer +1 -> Barber wakes up and gives haircut
   - When one customer comes, barber is notified and gives haircut.
3. More customer comes -> Waits if chair is empty
   - Next customer waits in chair in the waiting room when barber is busy
4. If chairs are full -> Customer leaves
   - If all chairs are full, then next customer leaves the shop

- Priority Scheduling Algorithm:

1. Process 1 comes -> starts getting executed
2. Next process comes with lower priority -> Waits till execution finishes
3. Another process comes with higher priority -> Pre-empts running process and starts executing
4. If two process have similar priority -> process that arrives first, gets executed

8. Have you made minimum 5 revisions of solution on GitHub?
- Yes, I have

GitHub Link: https://github.com/kindavishal/OS_Assignment/