OPERATING SYSTEM ASSIGNMENT

CA – 3 (Simulation Based)

Name	**Kasi Viswanath Pasala**
Section	**K18GT**
Regd No	**11806920**
Roll No	**A13**

Submitted to : Priyanka Mittal

PProblems:

<u>Q13:</u> A barbershop consists of a waiting room with n chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customer.

Code snippet for given question

```
#include<std
io.h>
#include<std
lib.h>
#include<un
istd.h>
#include<se
maphore.h>

#define N 5

time_t end_time;/*end time*/
sem_t mutex,customers,barbers;/*Three semaphors*/
int count=0;/*The number of customers waiting for haircuts*/

void
barber(void
*arg); void
customer(void
*arg);

int main(int argc,char *argv[])
{
   pthrea
   d_t
   id1,id
   2; int
   status
   =0;
   end_time=time(NULL)+20;/*Barber Shop Hours is 20s*/
```

```
    /*Semaphore
    initialization*/
    sem_init(&mute
    x,0,1);
    sem_init(&custo
    mers,0,0);
    sem_init(&barbe
    rs,0,1);

    /*Barber_thread initialization*/
    status=pthread_create(&id1,NULL,(void
    *)barber,NULL); if(status!=0)
            perror("create barbers is failure!\n");
    /*Customer_thread initialization*/
    status=pthread_create(&id2,NULL,(void
    *)customer,NULL); if(status!=0)
            perror("create customers is failure!\n");

    /*Customer_thread
    first blocked*/
    pthread_join(id2,NUL
    L);
    pthread_join(id1,NUL
    L);

    exit(0);
}
```

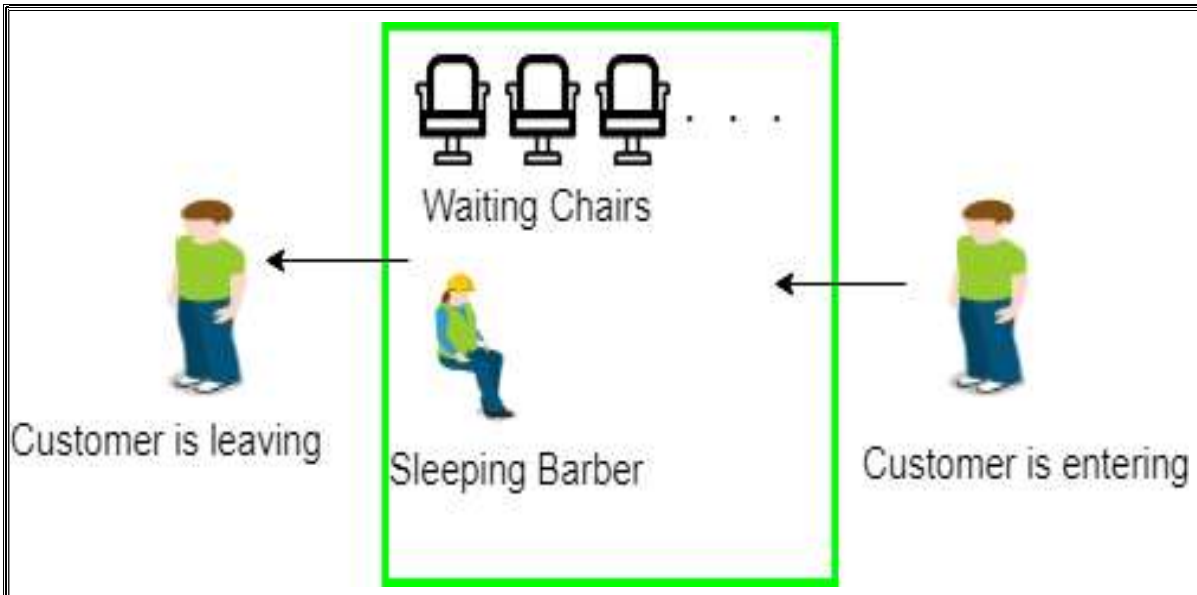
```
void barber(void *arg)/*Barber Process*/
{
   while(time(NULL)<end_time || count>0)
   {
            sem_wait(&customers);/*P(customers)*/
            sem_wait(&mutex);/*P(mutex)*/
            count--;
            printf("Barber:cut hair,count is:%d.\n",count);
            sem_post(&mutex);/*V(mutex)*/
            sem_post(&barbers);/*V(barbers)*/
            sleep(3);
   }
}

void customer(void *arg)/*Customers Process*/
{
   while(time(NULL)<end_time)
   {
            sem_wait(&mutex);/*P(mutex)*/
            if(count<N)
            {
                     count++;
                     printf("Customer:add count,count is:%d\n",count);
                     sem_post(&mutex);/*V(mutex)*/
                     sem_post(&customer);/*V(customers)*/
                     sem_wait(&barbers);/*P(barbers)*/
            }
            else
                     /*V(mutex)*/
                     /*If the number is full of customers,just put the mutex lock let go*/
                     sem_post(&mutex);
            sleep(1);
   }
}
```

Description of the problem in terms of operating system:

The analogy is based upon a hypothetical barber shop with one barber. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it.

Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves.

Based on a naïve analysis, the above description should ensure that the shop functions correctly, with the barber cutting the hair of anyone who arrives until there are no more customers, and then sleeping until the next customer arrives. In practice, there are a number of problems that can occur that are illustrative of general scheduling problems.



Explain all the test cases applied on the solution of barber sleeping problem?

Number	*Description*	*Input*	*Expected Output*
1	When there is no customer the barber will fall asleep and the customer will awake the barber.	• Chairs in waiting room: 4 • Chairs occupied: 0 • New customer enters: 1	• Barber is sleeping. • Customer wakes up the barber.
2	When the barber is busy and there is space for new customer in waiting room the customer will wait.	• Chairs in waiting room: 4 • Chairs occupied: 3 • New customer enters: 1	• Barber is busy. • New Customer entered. • Customer waits in waiting room.
3	When there is no space for new customer in waiting room the customer will wait.	• Chairs in waiting room: 4 • Chairs occupied: 4 • New customer enters: 1	• Barber is busy. • New Customer entered. • There is no free space customer leaves the shop.
4	When no new customer entersalso there is no customer in the shop then the barber will fall asleep.	• Chairs in waiting room: 4 • Chairs occupied: 0 • New customer enters: 0	• Barber is sleeping. • No new customer entered.

Complexity of above code: O (1), because there is no use of looping statements.

Algorithm Used:

1) The program takes the input from the user.
2) Thread concept is used.

3) **Complexity of above code:** O (1), because there is no use of looping statements.

Explain the Constraints applied in the code?

1) If there is no customer, the barber is sleeping and busy otherwise.
2) If there is an empty space in the waiting room the customer will wait and leave the shop otherwise.
3) If the barber is sleeping the customer will awake the barber.

Explain the boundary conditions of the implemented code?

A customer may arrive and observe that the barber is cutting hair, so he goes to the waiting room. While he is on his way, the barber finishes the haircut he is doing and goes to check the waiting room. Since there is no one there (the customer not having arrived yet), he goes back to his chair and sleeps. The barber is now waiting for a customer and the customer is waiting for the barber.

à Two customers may arrive at the same time when there happens to be a single seat in the waiting room. They observe that the barber is cutting hair, go to the waiting room, and both attempt to

occupy the single chair.

Have you made minimum five revisions ?

Yes, I have made 5 revisions.

Q16: Design a scheduling program that is capable of scheduling many processes that comes in at some time interval and are allocated the CPU not more than 10 time units. CPU must schedule processes having short execution time first. CPU is idle for 3 time units and does not entertain any process prior

this time. Scheduler must maintain a queue that keeps the order of execution of all the processes. Compute average waiting and turnaround time.

Solution: The above question is based on scheduling processes on the basis of their burst time i.e. the execution time. The process having the shortest execution time will be executed first. This program gives the average waiting and turnaround time.
For solving this problem I've created the program which is asking first for the number of processes and then for the arriving time and burst time for each process. On the basis of that the program helps in calculating the average turnaround and waiting time.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
```

```
int main()
{

  int count,i,j,m=0,n,y=0,time,remain=0,min,flag=0;
  int wait_time=0,turn_a_time=0,a_time[10],b_time[10],p[10],z[10];
  float k=0,x=0;
  printf("Enter number of Process:\t ");
  scanf("%d",&n);
  printf("\n\tArrival time should be greater than 2 as CPU remains idle for first 3 secs.\n");
  printf("\n\tBurst time should be less tha 10\n");
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
    scanf("%d",&a_time[count]);
    scanf("%d",&b_time[count]);
  }

 for(i=0;i<n;i++)
 {
          if(a_time[i]==0)
          {
                  printf("\nS INVALID ARRIVAL TIME\n");
                  getch();
                  exit(1);
          }
 }
 for(i=0;i<n;i++)
 {
          if(a_time[i]<3)
          {
                  printf("\nS INVALID Arrival Time it should be greater than 3\n");
                  getch();
                  exit(1);
          }
 }
   printf("\n\n\tProcess\t|Turnaround Time|Waiting Time\n\n");
   printf("\t=============================================\n");
 for(i=0;i<n;i++)
 {
          m=m+b_time[i];

min=m; time=m;
for(i=0;i<n;i++)
 {
          if(a_time[i]<time)
          {
                  time=a_time[i];
          }
 }
 for(i=time;i<=m;i=i+b_time[j])
 {
          min=m;
          remain=0;
          flag=0;

          for(count=0;count<n;count++)
          {

                  if(a_time[count]<=i)
                  {
```

```
                        if(b_time[count]<min)
                        {

                                min = b_time[count];
                                j=count;
                                flag=1;
                        }
                        remain=1;
                }
        }
        if(flag==1&&remain==1)
        {
                wait_time=i-a_time[j];
                turn_a_time=wait_time+b_time[j];
                printf("\tP[%d]\t|\t%d\t|\t%d\n",j+1,turn_a_time,wait_time);
                k=k+wait_time;
                x=x+turn_a_time;

                a_time[j]=m+1;
                p[y]=j+1;
                z[y]=i;
                y++;
        }
}
printf("\n\nAverage Waiting Time= %.2f\n",k/n);
printf("Avg Turnaround Time = %.2f",x/n);
printf("\n\n\nTotal time taken by processor to complete all the jobs : %d",m);
printf("\n\nQueue for order of execution:\n");
printf("\n\nProcess                 ");

for(i=0;i<n;i++)
{
        printf(" P[%d] ",p[i]);
        if(i==(n-1))
        {
                printf("End");
        }
}


  return 0;
  }
```

Algorithms used:

1) Arrival Time,
2) Completion time,
3) Turn Around Time,
4) Waiting Time

Explain the boundary conditions of the implemented code?

Arrival time should not be greater than 3.

Have you made minimum five revisions ?

Yes, I have made 5 revisions.