Task 5: Coding

2) Sleeping Barber Problem:

First of all we would create a waiting room which will hold customers. This can be done by first specifing the number of chairs available, and then creating an array of the chair's size with the name waiting room.

```
typedef int customer;
#define CHAIR_SIZE 5
```

The waiting room will be manipulated with two functions customer_enter() called by customer thread(s) and barber_finish(), which will be called by the barber thread.

```
/* The waiting room */
customer waiting_room[CHAIR_SIZE];

int customer_enter(customer cust)
{
/* If the barber is sleeping, then wake him, otherwise if waiting_room not
full then sit and return 0. */
/* the barber thread id is stored as global variable, it will be used here
to check if the barber thread is sleeping or not.
/* If the waiting room is full then leave, and return -1 */
}

int barber_finish(customer *cust)
{
/* If more customers are waiting, then take the next customer and return 0.
*/ /* If no customers are waiting, then return -1 */
}
```

The main function will initialize the mutex and the empty and full semaphore for tracking the number of chairs available. It will also create the barber thread and the customer(s) threads, and then sleep for a period of time, and then exit.

```
int main(int argc, char *argv[])
{
/* 1. Get command line arguments argv[1],argv[2] */
/* argv[1]: Sleep time (in seconds) in total to simulate the running of the program
*/
/* argv[2]: Number of customers */
/* 2. Initialize buffer */
/* 3. Create barber thread */
/* 4. Create customer thread(s) */
/* 5. Sleep */
/* 6. Exit */
}
```

Now the barber will cut hair, by taking a customer from the waiting room, if the barber is asleep, then he will automatically be woken up by the customer and then proceed with the customer from waiting room.
The customer will come and sit in waiting area if there is space, otherwise he will leave.

```
void *customer(void *param)
{
customer cust;
while (TRUE)
{
/* sleep for a random period of time */
/* to simulate random arrival times */
sleep(...);
/* generate a random number to represent a customer */
cust = rand();
if (customer_enter(item))
fprintf("customer %d left because all chairs were occupied\n", cust);
else
printf("Customer waits for barber. %d\n", cust);
}
}
```

```c
void *barber(void *param)
{
customer cust;
while (TRUE)
{
/* sleep for a random period of time */
/* to simulate time taken to cut hair */
sleep(...);
if (barber_finish(&cust))
sleep(INT_MAX); /* barber sleeps until woken by customer*/
else
printf("Customer's hair is cut %d\n", cust);
}
}
```

In this problem we will use semaphore for the empty seats and the full seats just like Producer-Consumer problem, we may also want to use additional semaphore to keep track if the barber is sleeping. We also keep track of the thread the barber will be running as a global variable, so that the customer can wake up the barber thread.