Operating System (CS301)
Assignment - 8
U19CS012

1. The Synchronization problem called **Sleeping Barber** is described as follows:

- ✓ A barber shop has a **single** barber, a single barber's chair in a small room, and a large waiting room with **n** seats.
- ✓ After servicing one customer, the barber checks whether any customers are waiting in the waiting room. If so, he admits one of them and starts serving him; otherwise, he goes to sleep in the barber's chair.
- ✓ A customer enters the waiting room only if there is at least one vacant seat and either waits for the barber to call him if the barber is busy, or wakes the barber if he is asleep.

Identify the synchronization requirements between the barber and customer processes.

When the Barber wants to address the Person in Waiting Queue and Another Customer enters the shop, then the waiting_customer Queue should be updated appropriately.

Common Header File {Barber Class}

```
// This is General Header File that Contains the Barber Class
#include <unistd.h>
#include <iostream>
#include <queue>
#include <thread>
#include <mutex>

using namespace std;
// Waiting Room of 'n' Seats
struct WaitingRoom
{
    int TotalSeats;
    queue<int> Customers;
};
```

```
// Defination of Barber Class
class Barber
{

private:
    // ID of Barber
    int id;

public:
    // Default Constructor
    Barber() {}

    // Constructor with ID
    Barber(int id)
    {
        this->id = id;
    }

    // Member Function for Hair Cutting Task {itrs -> Iterations}
    void doTask(WaitingRoom &room, mutex &mtx, mutex &stdOutMtx, int &itrs)
    {

        while (true)
        {
            mtx.lock();

            // If Number of Iterations are Zero and No Customer in Waiting Room
            if (itrs == -1 && room.Customers.empty())
            {
                mtx.unlock();
                break;
            }

            // If there is Customer in Waiting Room
            if (!room.Customers.empty())
            {
                // The Customer who Came First, Should be Served First
                int toBeServed = room.Customers.front();
                room.Customers.pop();

                mtx.unlock();
                sleep(1);

                // Mutex to Print the Output {Standarr Output}
                stdOutMtx.lock();
                cout << "Barber: " << this->id << " has done the task for the customer : " <<
toBeServed << endl;
                stdOutMtx.unlock();
            }
```

```
            else
            {
                mtx.unlock();
                sleep(1);
            }
        }
    }
};
```

a.) **Code** the barber and customer processes such that deadlocks do not arise.

```
// Include the Class Defined in "barberheader" Header File
#include "barberheader.hpp"
#include <thread>

// Maximum Number of Iterations
#define ITR 15
// Maximum Number of Random Customers
#define MAX_RAND_CUSTOMERS 7

int main()
{
    int size;
    cout << "Enter the size for the Waiting Room : ";
    cin >> size;

    cout << "~~~~~~~~~~~~~~~~~Initializing~~~~~~~~~~~~~~~~~" << endl;

    // Mutex for Handling Mutual Exclusion Problem in Sleeping Barber
    mutex mtx;
    // Mutex of Std Output {Print on Screen}
    mutex stdOutMtx;

    Barber barber(0);
    WaitingRoom room({size, queue<int>()});

    int itr = ITR;

    // Create a Thread
    thread barberThread(&Barber::doTask, ref(barber), ref(room), ref(mtx), ref(stdOutMtx),
ref(itr));

    // Intially, there are no Customer in Shop
    int customer = 0;

    int temp = 0, newCustomers;
```

```
    // Randomly Customers will be Coming for itr iterations
    while (itr--)
    {
        newCustomers = rand() % MAX_RAND_CUSTOMERS;
        newCustomers = max(0, newCustomers);

        if (newCustomers > 0)
        {
            temp = 0;

            mtx.lock();
            // Add the Customer to Waiting Queue {If Possible}
            while (newCustomers-- && room.Customers.size() < room.TotalSeats)
            {
                room.Customers.push(customer++);
                temp++;
            }
            mtx.unlock();
        }

        // Print the Output on Screen
        stdOutMtx.lock();
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
        cout << "Iteration Number is : " << itr << endl;
        cout << "Customers Added are : " << temp << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
        stdOutMtx.unlock();

        sleep(1);
    }

    barberThread.join();

    cout << "~~~~~~~~~~~~~~~~~Task Completed~~~~~~~~~~~~~~~~~" << endl;

    return 0;
}
```

Output [Colcalc {Linux Environment}]

```
~
~$ g++ singleBarber.cpp -I . -pthread -o singleBarber.exe
~$ ./singleBarber.exe
Enter the size for the Waiting Room : 9
~~~~~~~~~~~~~~~~~~~~Initializing~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 14
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 13
Customers Added are : 4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 12
Customers Added are : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 11
Customers Added are : 5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 10
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 9
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 4
Barber: 0 has done the task for the customer : 5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 8
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 6
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 7
Customers Added are : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 7
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 6
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 8
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 5
Customers Added are : 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 9
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 4
Customers Added are : 1
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 10
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 3
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 11
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 2
Customers Added are : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 12
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 1
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 13
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 0
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 14
Barber: 0 has done the task for the customer : 15
Barber: 0 has done the task for the customer : 16
Barber: 0 has done the task for the customer : 17
Barber: 0 has done the task for the customer : 18
Barber: 0 has done the task for the customer : 19
Barber: 0 has done the task for the customer : 20
Barber: 0 has done the task for the customer : 21
Barber: 0 has done the task for the customer : 22
Barber: 0 has done the task for the customer : 23
~~~~~~~~~~~~~~~~~~~~Task Completed~~~~~~~~~~~~~~~~~~~~
```

b.) Consider the Sleeping-Barber Problem with the modification that there are k barbers and k barber chairs in the barber room, instead of just one. Write a program to coordinate the barbers and the customers.

Code

```
// Include the Class Defined in "barberheader" Header File
#include "barberheader.hpp"
#include <thread>

// Maximum Number of Iterations
#define ITR 15
// Maximum Number of Random Customers
#define MAX_RAND_CUSTOMERS 7

// Same Code from Single Barber is Modified for Multiple Barbers

int main()
{
    int size, barberCount;

    cout << "Enter the size for Waiting Room : ";
    cin >> size;

    cout << "Enter the number of Barbers     : ";
    cin >> barberCount;

    cout << "~~~~~~~~~~~~~~~~~~Initializing~~~~~~~~~~~~~~~~~~" << endl;

    // Mutex for Handling Mutual Exclusion Problem in Sleeping Barber
    mutex mtx;
    // Mutex of Std Output {Print on Screen}
    mutex stdOutMtx;

    WaitingRoom room({size, queue<int>()});

    int itr = ITR;

    Barber barbers[barberCount];

    // Create 1 Thread for Each Barber
    thread barberThreads[barberCount];

    for (int i = 0; i < barberCount; i++)
    {
        barbers[i] = Barber(i);
        barberThreads[i] = thread(&Barber::doTask, ref(barbers[i]), ref(room), ref(mtx),
ref(stdOutMtx), ref(itr));
```

```
    }

    int customer = 0;
    int temp = 0;
    int newCustomers;

    // Randomly Customers will be Coming for itr iterations
    while (itr--)
    {
        newCustomers = rand() % MAX_RAND_CUSTOMERS;
        newCustomers = max(0, newCustomers);

        if (newCustomers > 0)
        {
            temp = 0;

            mtx.lock();
            while (newCustomers-- && room.Customers.size() < room.TotalSeats)
            {
                room.Customers.push(customer++);
                temp++;
            }
            mtx.unlock();
        }

        // Print the Output on Screen
        stdOutMtx.lock();
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
        cout << "Iteration Number is : " << itr << endl;
        cout << "Customers Added are : " << temp << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
        stdOutMtx.unlock();

        sleep(1);
    }

    for (int i = 0; i < barberCount; i++)
        barberThreads[i].join();

    cout << "~~~~~~~~~~~~~~~~~~Task Completed~~~~~~~~~~~~~~~~~~" << endl;

    return 0;
}
```

Output

```
~$ g++ multipleBarbers.cpp -I . -pthread -o multipleBarbers.exe
~$ ./multipleBarbers.exe
Enter the size for Waiting Room : 9
Enter the number of Barbers     : 4
~~~~~~~~~~~~~~~~~~~~~~Initializing~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 14
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 13
Customers Added are : 4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 12
Customers Added are : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 1
Barber: 2 has done the task for the customer : 2
Barber: 3 has done the task for the customer : 3
Barber: 1 has done the task for the customer : 4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 11
Customers Added are : 5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 5
Barber: 2 has done the task for the customer : 6
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 10
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 7
Barber: 1 has done the task for the customer : 8
Barber: 3 has done the task for the customer : 9
Barber: 2 has done the task for the customer : 10

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 9
Customers Added are : 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 11
Barber: 1 has done the task for the customer : 12
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 8
Customers Added are : 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 13
Barber: 1 has done the task for the customer : 14
Barber: 3 has done the task for the customer : 15
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 7
Customers Added are : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 16
Barber: 1 has done the task for the customer : 17
Barber: 3 has done the task for the customer : 18
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 6
Customers Added are : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 19
Barber: 1 has done the task for the customer : 20
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 5
Customers Added are : 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 21
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 4
Customers Added are : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 22
Barber: 2 has done the task for the customer : 23
Barber: 1 has done the task for the customer : 24
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 3
Customers Added are : 5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 25
Barber: 2 has done the task for the customer : 26
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 2
Customers Added are : 6
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 27
Barber: 3 has done the task for the customer : 28
Barber: 2 has done the task for the customer : 29
Barber: 1 has done the task for the customer : 30

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 1
Customers Added are : 4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 31
Barber: 3 has done the task for the customer : 32
Barber: 1 has done the task for the customer : 34
Barber: 2 has done the task for the customer : 33
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Iteration Number is : 0
Customers Added are : 6
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Barber: 0 has done the task for the customer : 35
Barber: 3 has done the task for the customer : 36
Barber: 1 has done the task for the customer : 37
Barber: 2 has done the task for the customer : 38
Barber: 0 has done the task for the customer : 39
Barber: 1 has done the task for the customer : 41
Barber: 3 has done the task for the customer : 40
Barber: 2 has done the task for the customer : 42
Barber: 0 has done the task for the customer : 43
Barber: 1 has done the task for the customer : 44
Barber: 3 has done the task for the customer : 45
Barber: 2 has done the task for the customer : 46
Barber: 0 has done the task for the customer : 47
~~~~~~~~~~~~~~~~~~~~Task Completed~~~~~~~~~~~~~~~~~~~~
```

SUBMITTED BY: **U19CS012** BHAGYA VINOD RANA