OS LAB: ASSIGNMENT 5
REPORT

Usage of Semaphores to synchronize between threads

Group number: 30

Group members

Akash Das	20CS10006
Prakhar Singh	20CS10045
Rohit Kumar Prajapati	20CS30041
Saras Umakant Pantulwar	20CS30046

Data Structures:

1. GUEST DETAILS:

```
vector<pair<int, int>> guest_id // (guest_id, guest_priority)
```

2. ROOM DETAILS:

```
typedef struct Room
{
    int room_id;    // room id
    int guest_id;   // guest_id (0 if no one is occupying the room)
    int guest_priority;     // priority of the guest
    int dirty;   // how dirty is the room
    int time;   // time for which the room was occupied
    sem_t room_sem;  // semaphore used for each room
    Room(int id, int g, int p)  // constructor
    {
        room_id = id;
        guest_id = g;
        guest_priority = p;
        dirty = 0;
        time = 0;
        sem_init(&room_sem, 0, 1);
    }
} Room;
```

3. STAFF_ID:

```
vector<int> staff_id; // staff_id's were stored
```

4. GLOBAL VARIABLES:
   a. `int dirty_level = 0; //  current dirty level of the hotel`
   b. `vector<int> rooms_to_be_cleaned; // room's that needs to be cleaned`
   c. `sem_t room_list_sem; // to lock the room_list`
   d. `sem_t cleaning_sem; // to lock the cleaning_staff`
   e. `sem_t vec_sem; // semaphore to access rooms_to_be_cleaned vector`

USES OF SEMAPHORES:

1. Lock each room with a value = 1, so that only one can access the room at a time.

```
sem_init(&room_sem, 0, 1);
```

2. Lock the vector of rooms to a limit of the Number of guests so that at max that amount of threads can access the room details.

```
sem_init(&room_list_sem, 0, num_guests);
```

3. Lock Cleaning staff vector in the Hotel

```
sem_init(&cleaning_sem, 0, num_cleaners);
```

4. Lock rooms_to_be_cleaned vector

```
sem_init(&vec_sem, 0, 1);
```

Additional Information:

1. we have used mutex for stdout: **pthread_mutex_init(&std_lock, NULL);**
2. we have used the **SIGQUIT** signal to wake/remove guests from the room in case of cleaning or a higher priority guest waits for the room.

DESIGN:

- Guest:
    - the guest threads wait for rooms to be cleaned if any rooms are under the cleaning process by searching **sem_getvalue** of cleaning_sem.
    - it then searches for an empty room if found, it then occupies the room for a random stay_time.
    - if it doesn't find a room it then kicks any guest with the lower/lowest priority, if not found it goes back to sleep again
    - after stay_time it wakes up leaves the room (**sem_post**), and then goes back to sleep again.
- Staff:
    - it waits for rooms to get dirty and also updates the vector that contains all the room ids that need to be cleaned and select a random room out of them.
    - it then locks the room that needs cleaning and after cleaning it releases the room (**sem_post**), marks the room as clean, and goes back to search for another room that needs cleaning
    - after cleaning is done then only guests will be allowed to stay in the rooms again.