CS307 PA3

Synchronization Sightseeing: Managing a Tourist Attraction with Semaphores

NAME: UTKU

SURNAME: GENC

SUID: 30611

CONTENTS

1- What Is The Point Of This PA3?

In this homework, subject to be tought is “***Semaphores***”. While solving the problem, what is expected from us is to use this synchronization primitive and gain a deeper understanding about it.

2-What Is Semaphore?

It is a basic tool for us to use for achieving synchronization. Lets say we have multiple programs, or threads. And we want to organize them in such a way that only **“n”** of them enters the **critical part** of our code. In this case, we use semaphores.

It is actually very similar to mutex, at least in terms of basic idea behind it. When we initialize a semaphore, we also assign it a value: N. When a thread wants to enter the critical part, it **decrement** the value of N by 1. If the value is bigger than zero, it successfully enter. If it is zero, the thread becomes blocked until another thread **wakes it up** by raising **a signal**. Also, when a thread raise a signal, the value **incremented** by one.

It is not that confusing, but we can represent it even more **simpler** (And more ***humanly***, sometimes we forget it and make it boring as hell): Imagine a bridge that can only carry N cars, and at the beginning of this bridge there is a **police officer** that counts the cars, making sure N number constraint is satisfied. When a car exits the bridge, a signal is activated and police officer takes inside one more car from the queue. In this case, police officer is semaphore, and the cars are threads.

We have 3 basic methods to use:

1- Sem_init()
2- Sem_post()
3- Sem_wait()

Sem_init allows us to initialize our semaphore. **Sem_post** is increment the value, and also create a signal. **Sem_wait** is decrementing the value, and if there is **no room in the bridge**, it simply waits for a signal.

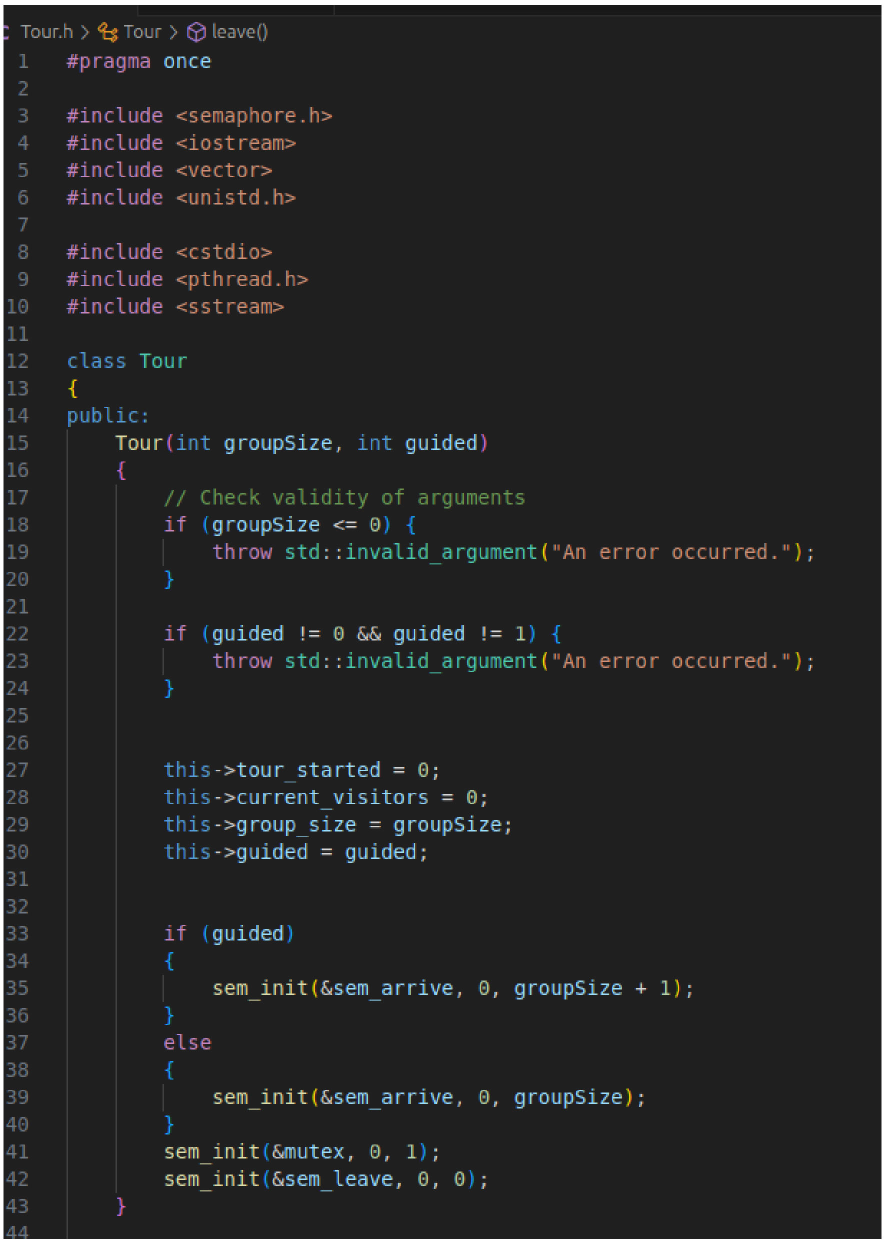
3-How Did I Use Semaphores to Solve?

In the question, it is asked us to implement 3 function including constructor: **Tour(int** groupSize, **int** guided**)**, **arrive()**, **leave()**. Of course i am not going to explain the details of the question, because you can always read the 16 pages homework document, but long story short we need to synchronize the functions so that intended input can be realized.

Our first problem is only limited visitors can be present at the same time in the attraction site. Solution of this problem is pretty easy actually. We use simple semaphore with the value **groupSize**, so that ensuring only **groupSize** of people can enter. When a visitor enter, it decrements and when someone leaves the value is incremented.

The second problem is we need to make sure that when a tour is finished, new visitors can only enter **after the last visitor make the announcement.** First we need to make sure a thread is or is not the last visitor. And if it is, it should give enough signals (groupSize times) so that new visitors can enter the site.

The last is our synchronization problem. And i solve this using **“barrier”** and **“turnstile”** algorithm which i read from here: click, page 49. You can learn more about these two in that pdf. Basically what i do is constracting a barrier, which makes sure that if a thread finish travelling earlier than the guide, it should wait. And when tour guide announce that the tour is over, it also give **one signal, and waking up one of the waiters.** And than this thread **wakes another one** until all of the visitors leave. If the tour guide is the first one who finishes, no problem! This time semaphore value becomes one, so that a “normally it should wait” thread can continue. At the end, the semaphore value becomes one, which is unusable. So that last visitor should decrement its value one more time, ensuring it is equal to zero. Below, i share my codes with detailed comments. If you look at them carefully, it is not that hard to understand.

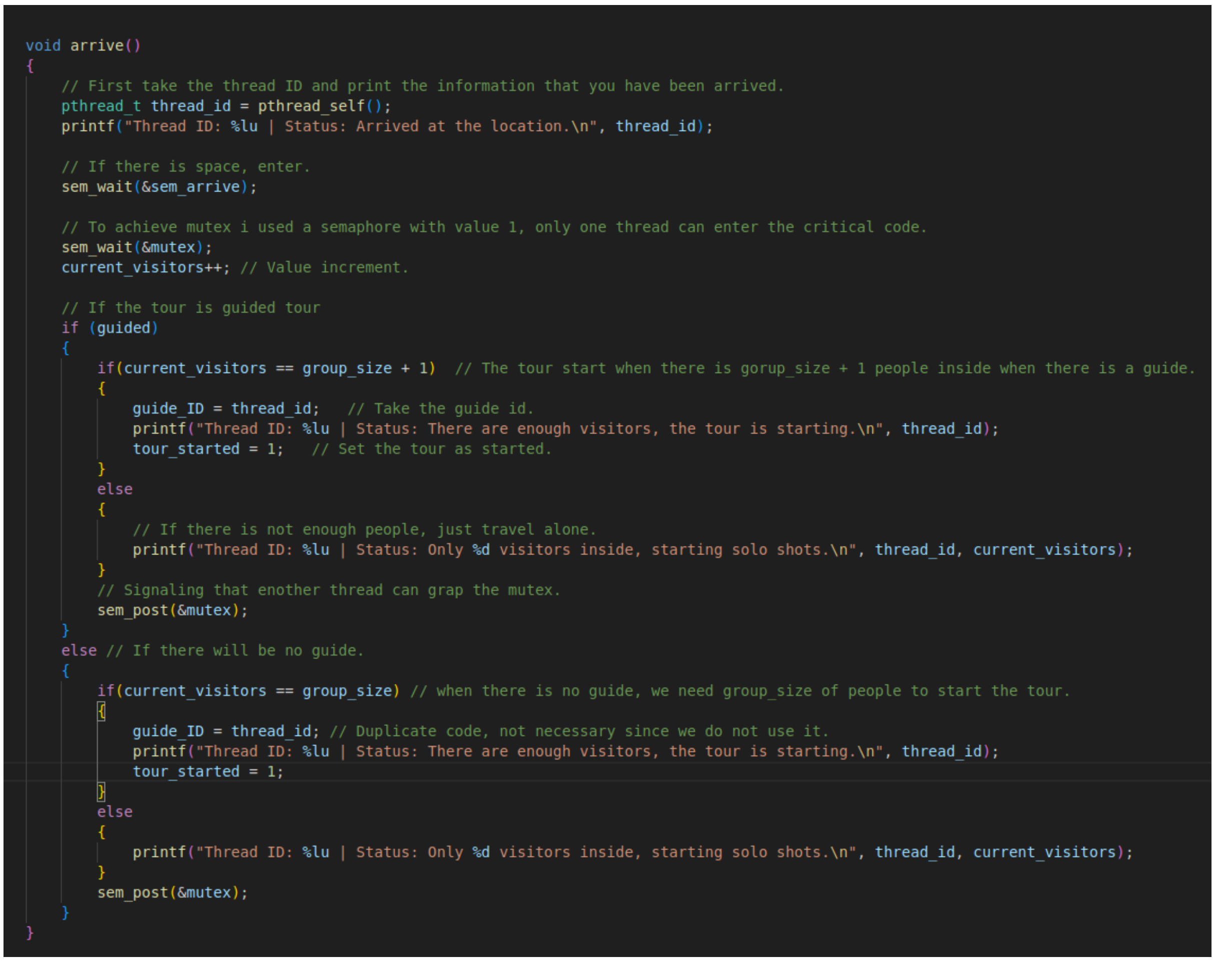


```cpp
#pragma once

#include <semaphore.h>
#include <iostream>
#include <vector>
#include <unistd.h>

#include <cstdio>
#include <pthread.h>
#include <sstream>

class Tour
{
public:
    Tour(int groupSize, int guided)
    {
        // Check validity of arguments
        if (groupSize <= 0) {
            throw std::invalid_argument("An error occurred.");
        }

        if (guided != 0 && guided != 1) {
            throw std::invalid_argument("An error occurred.");
        }


        this->tour_started = 0;
        this->current_visitors = 0;
        this->group_size = groupSize;
        this->guided = guided;


        if (guided)
        {
            sem_init(&sem_arrive, 0, groupSize + 1);
        }
        else
        {
            sem_init(&sem_arrive, 0, groupSize);
        }
        sem_init(&mutex, 0, 1);
        sem_init(&sem_leave, 0, 0);
    }
```

The first function is constructor. It takes two values and initialize all the variables, including semaphores. I used 3 semaphore:

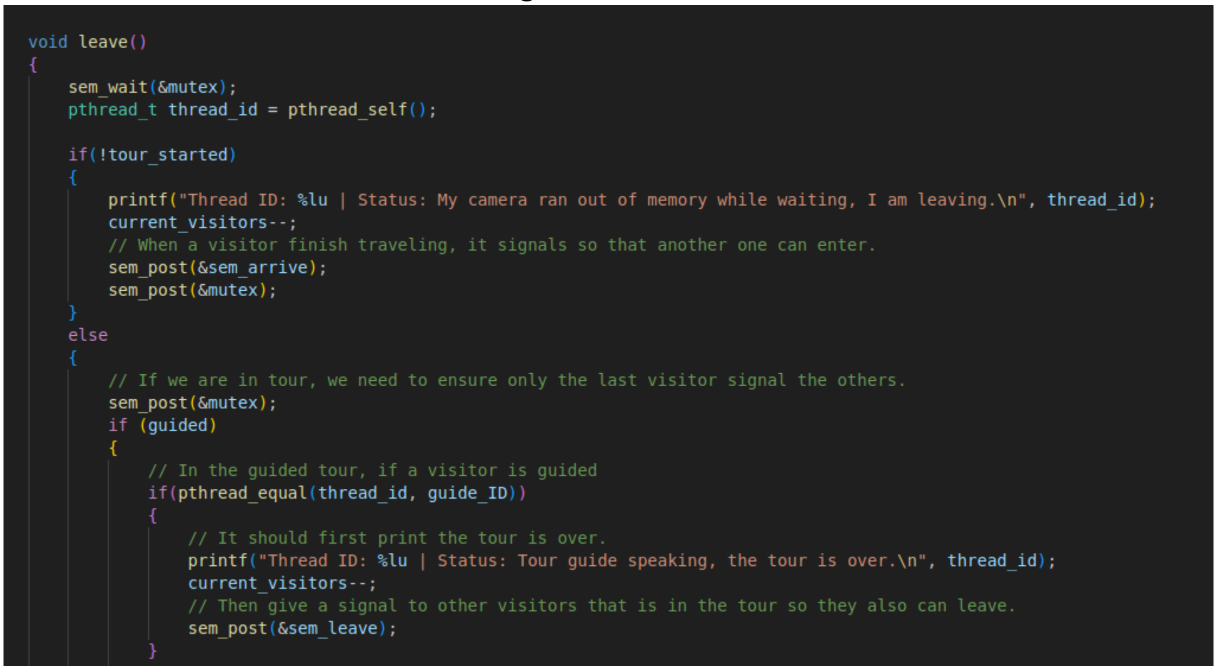
1- Sem_arrive
2- Sem_leave
3- Mutex

Sem_arrive is used for ensuring a limited number of people enters the site. Sum_leave is used for ensuring synchronization when visitors are leaving (barrier), and mutex is used for critical parts of the code. As you can see, leave value is zero, which means no thread is allowed until a signal comes. Also mutex value is one, ensuring only one thread can execute critical parts. The constructor also throws an exception when group size is negative and guided is not equals to zero and one.



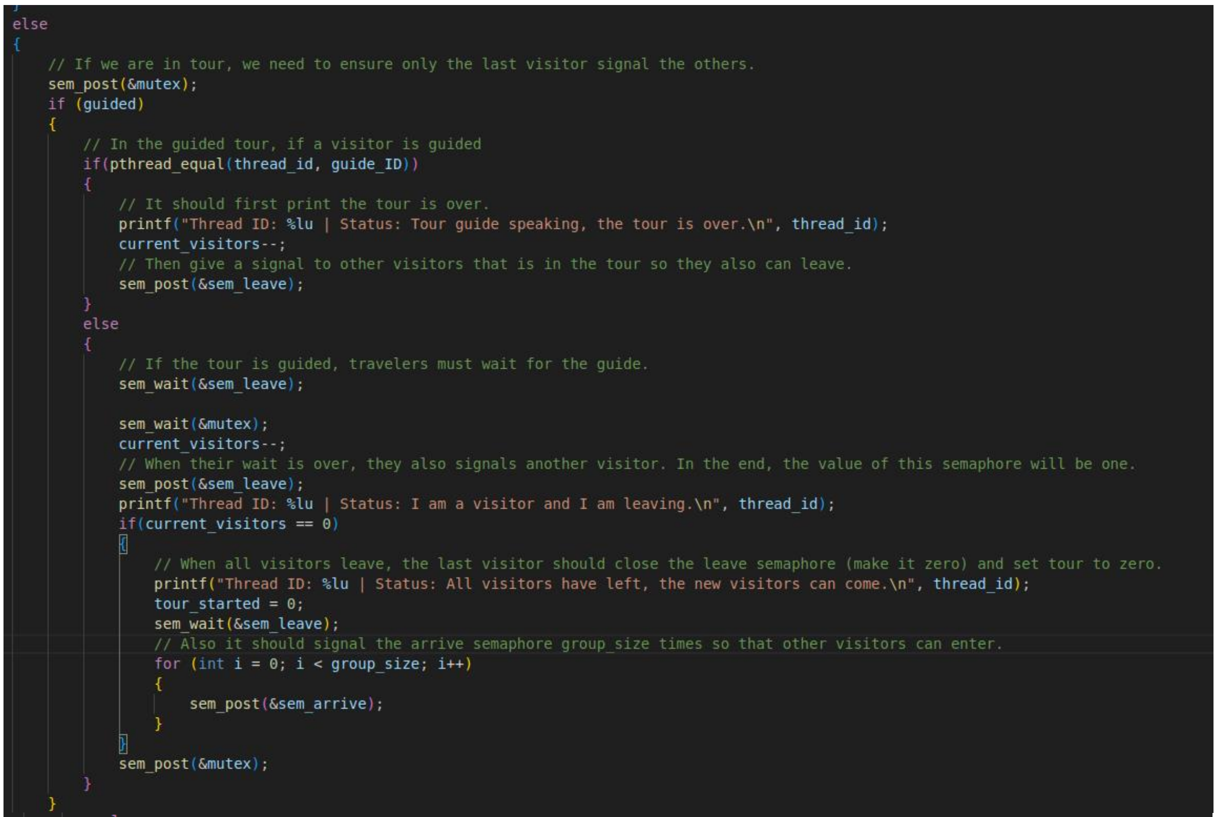
```c
void arrive()
{
    // First take the thread ID and print the information that you have been arrived.
    pthread_t thread_id = pthread_self();
    printf("Thread ID: %lu | Status: Arrived at the location.\n", thread_id);

    // If there is space, enter.
    sem_wait(&sem_arrive);

    // To achieve mutex i used a semaphore with value 1, only one thread can enter the critical code.
    sem_wait(&mutex);
    current_visitors++; // Value increment.

    // If the tour is guided tour
    if (guided)
    {
        if(current_visitors == group_size + 1)  // The tour start when there is gorup_size + 1 people inside when there is a guide.
        {
            guide_ID = thread_id;   // Take the guide id.
            printf("Thread ID: %lu | Status: There are enough visitors, the tour is starting.\n", thread_id);
            tour_started = 1;   // Set the tour as started.
        }
        else
        {
            // If there is not enough people, just travel alone.
            printf("Thread ID: %lu | Status: Only %d visitors inside, starting solo shots.\n", thread_id, current_visitors);
        }
        // Signaling that enother thread can grap the mutex.
        sem_post(&mutex);
    }
    else // If there will be no guide.
    {
        if(current_visitors == group_size) // when there is no guide, we need group_size of people to start the tour.
        {
            guide_ID = thread_id; // Duplicate code, not necessary since we do not use it.
            printf("Thread ID: %lu | Status: There are enough visitors, the tour is starting.\n", thread_id);
            tour_started = 1;
        }
        else
        {
            printf("Thread ID: %lu | Status: Only %d visitors inside, starting solo shots.\n", thread_id, current_visitors);
        }
        sem_post(&mutex);
    }
}
```

In here you can see my arrive function. I am not going to explain again since the comments is clear enough.

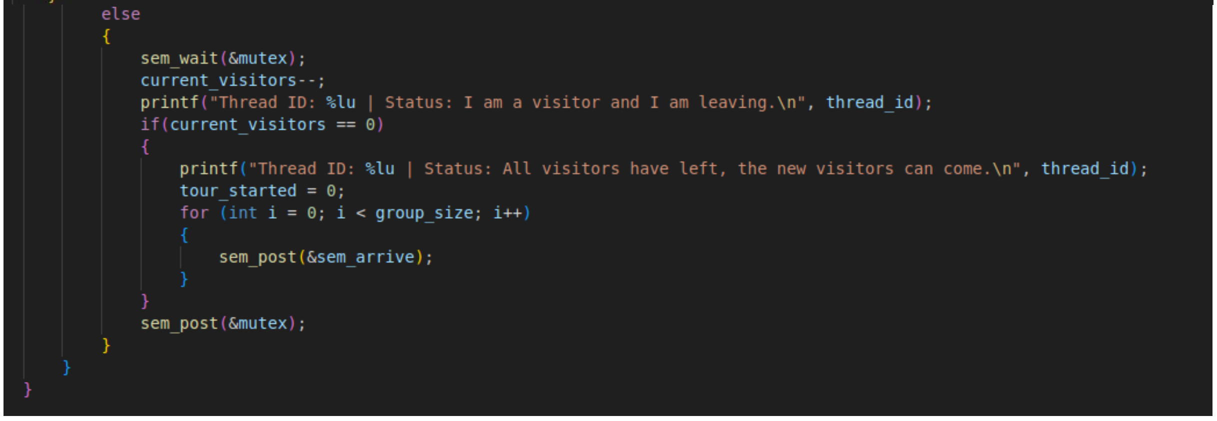


```c
void leave()
{
    sem_wait(&mutex);
    pthread_t thread_id = pthread_self();

    if(!tour_started)
    {
        printf("Thread ID: %lu | Status: My camera ran out of memory while waiting, I am leaving.\n", thread_id);
        current_visitors--;
        // When a visitor finish traveling, it signals so that another one can enter.
        sem_post(&sem_arrive);
        sem_post(&mutex);
    }
    else
    {
        // If we are in tour, we need to ensure only the last visitor signal the others.
        sem_post(&mutex);
        if (guided)
        {
            // In the guided tour, if a visitor is guided
            if(pthread_equal(thread_id, guide_ID))
            {
                // It should first print the tour is over.
                printf("Thread ID: %lu | Status: Tour guide speaking, the tour is over.\n", thread_id);
                current_visitors--;
                // Then give a signal to other visitors that is in the tour so they also can leave.
                sem_post(&sem_leave);
            }
```

This is the first part of the leave() function. I use mutexes to prevent data racings, which can be occur in here.

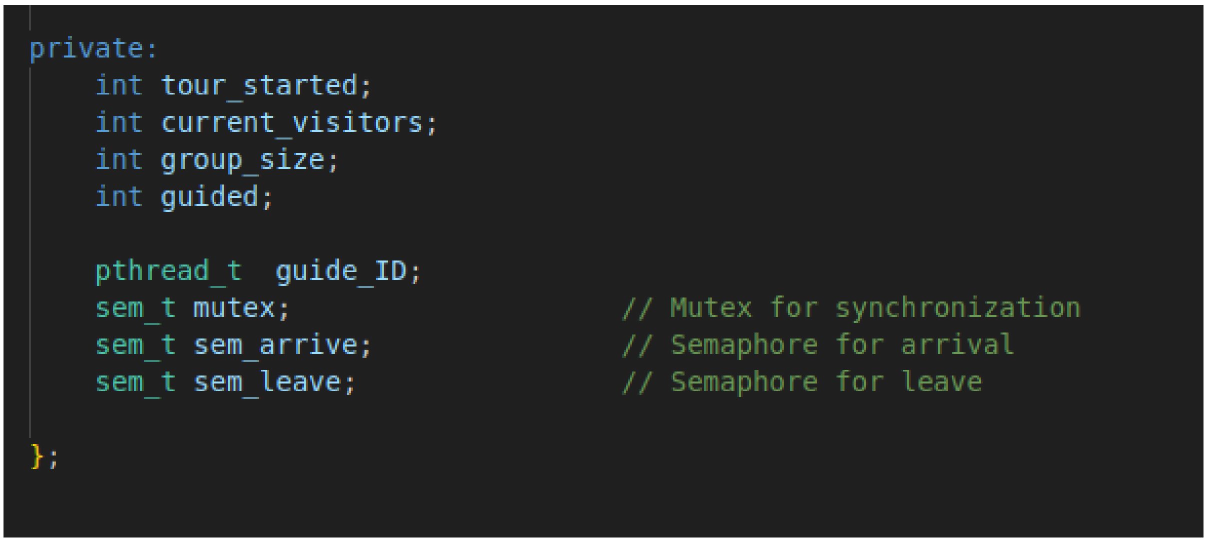


```c
else
{
    // If we are in tour, we need to ensure only the last visitor signal the others.
    sem_post(&mutex);
    if (guided)
    {
        // In the guided tour, if a visitor is guided
        if(pthread_equal(thread_id, guide_ID))
        {
            // It should first print the tour is over.
            printf("Thread ID: %lu | Status: Tour guide speaking, the tour is over.\n", thread_id);
            current_visitors--;
            // Then give a signal to other visitors that is in the tour so they also can leave.
            sem_post(&sem_leave);
        }
        else
        {
            // If the tour is guided, travelers must wait for the guide.
            sem_wait(&sem_leave);

            sem_wait(&mutex);
            current_visitors--;
            // When their wait is over, they also signals another visitor. In the end, the value of this semaphore will be one.
            sem_post(&sem_leave);
            printf("Thread ID: %lu | Status: I am a visitor and I am leaving.\n", thread_id);
            if(current_visitors == 0)
            {
                // When all visitors leave, the last visitor should close the leave semaphore (make it zero) and set tour to zero.
                printf("Thread ID: %lu | Status: All visitors have left, the new visitors can come.\n", thread_id);
                tour_started = 0;
                sem_wait(&sem_leave);
                // Also it should signal the arrive semaphore group_size times so that other visitors can enter.
                for (int i = 0; i < group_size; i++)
                {
                    sem_post(&sem_arrive);
                }
            }
            sem_post(&mutex);
        }
    }
```



```c
        else
        {
            sem_wait(&mutex);
            current_visitors--;
            printf("Thread ID: %lu | Status: I am a visitor and I am leaving.\n", thread_id);
            if(current_visitors == 0)
            {
                printf("Thread ID: %lu | Status: All visitors have left, the new visitors can come.\n", thread_id);
                tour_started = 0;
                for (int i = 0; i < group_size; i++)
                {
                    sem_post(&sem_arrive);
                }
            }
            sem_post(&mutex);
        }
    }
}
```

(Note: In here, “for loop group_size” for the guided tour must be group_size + 1, i realized it after i test the code later. Inside the code, i change it)

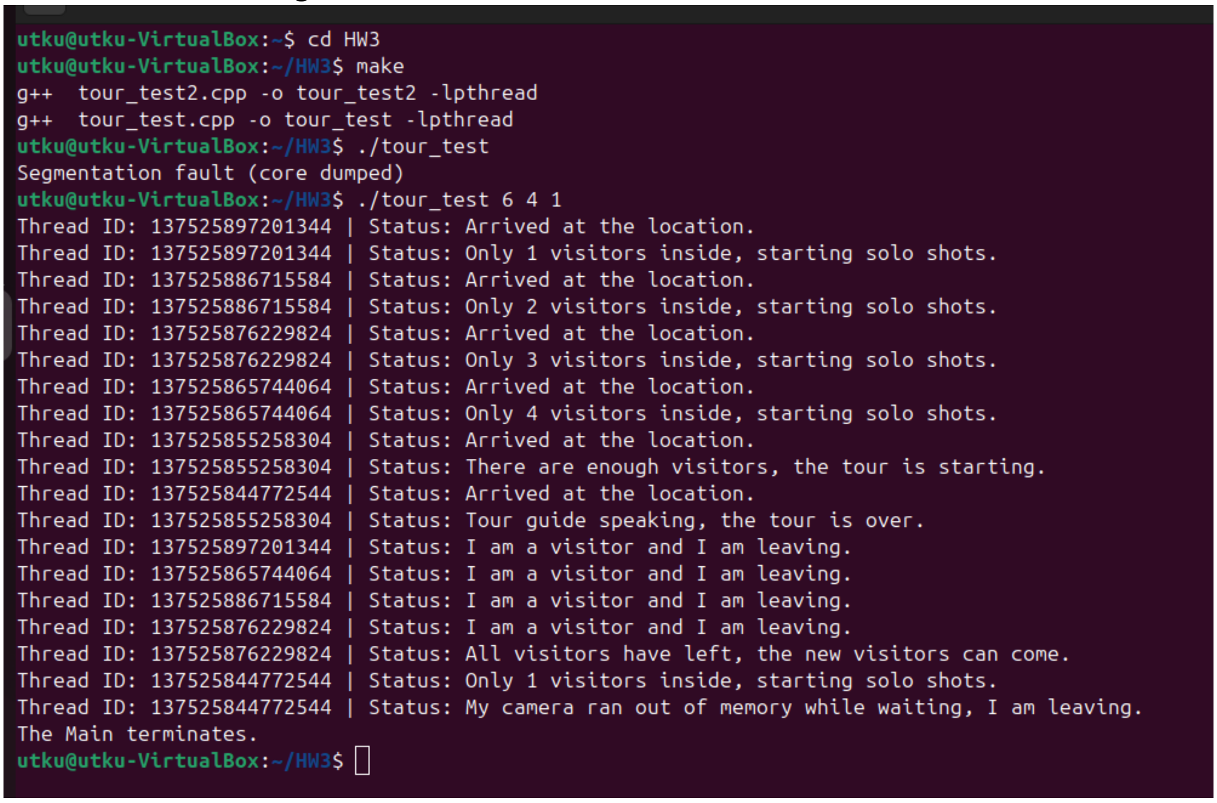


```
private:
    int tour_started;
    int current_visitors;
    int group_size;
    int guided;

    pthread_t  guide_ID;
    sem_t mutex;                        // Mutex for synchronization
    sem_t sem_arrive;                   // Semaphore for arrival
    sem_t sem_leave;                    // Semaphore for leave

};
```

In here, you can see the private fields of the Tour object.

4-Why Does it Work?

Altough it is very hard to prove by writing, some of the key points shows it is working indeed:

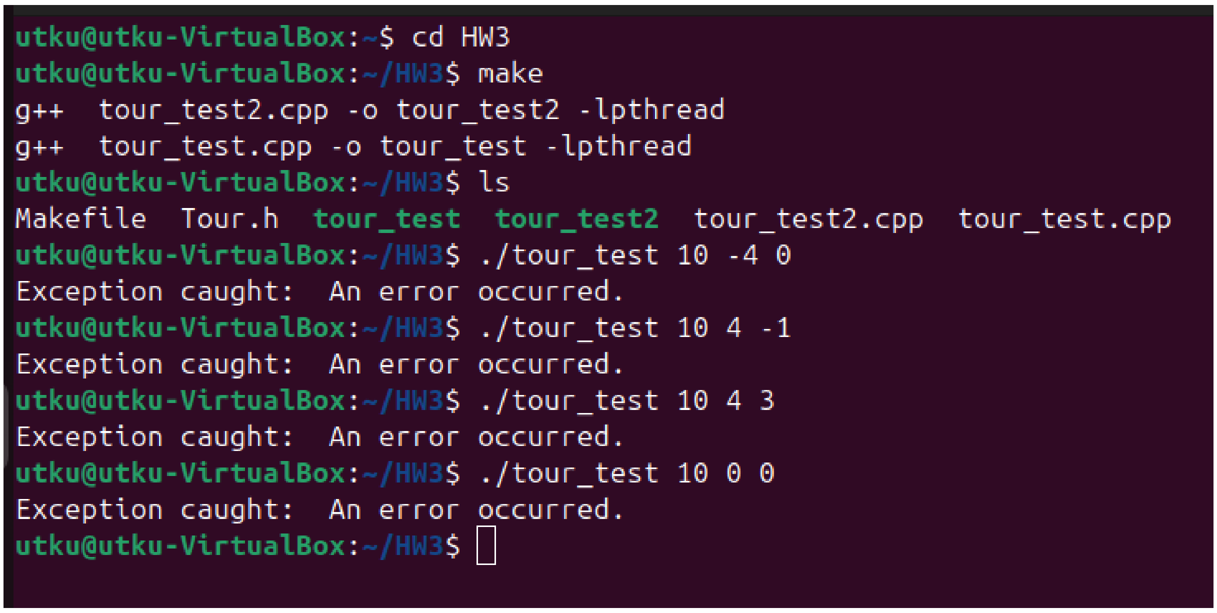


```
utku@utku-VirtualBox:~$ cd HW3
utku@utku-VirtualBox:~/HW3$ make
g++  tour_test2.cpp -o tour_test2 -lpthread
g++  tour_test.cpp -o tour_test -lpthread
utku@utku-VirtualBox:~/HW3$ ./tour_test
Segmentation fault (core dumped)
utku@utku-VirtualBox:~/HW3$ ./tour_test 6 4 1
Thread ID: 137525897201344 | Status: Arrived at the location.
Thread ID: 137525897201344 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 137525886715584 | Status: Arrived at the location.
Thread ID: 137525886715584 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 137525876229824 | Status: Arrived at the location.
Thread ID: 137525876229824 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 137525865744064 | Status: Arrived at the location.
Thread ID: 137525865744064 | Status: Only 4 visitors inside, starting solo shots.
Thread ID: 137525855258304 | Status: Arrived at the location.
Thread ID: 137525855258304 | Status: There are enough visitors, the tour is starting.
Thread ID: 137525844772544 | Status: Arrived at the location.
Thread ID: 137525855258304 | Status: Tour guide speaking, the tour is over.
Thread ID: 137525897201344 | Status: I am a visitor and I am leaving.
Thread ID: 137525865744064 | Status: I am a visitor and I am leaving.
Thread ID: 137525886715584 | Status: I am a visitor and I am leaving.
Thread ID: 137525876229824 | Status: I am a visitor and I am leaving.
Thread ID: 137525876229824 | Status: All visitors have left, the new visitors can come.
Thread ID: 137525844772544 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 137525844772544 | Status: My camera ran out of memory while waiting, I am leaving.
The Main terminates.
utku@utku-VirtualBox:~/HW3$
```

1- The arrive semaphore ensures that only N number of people can visit the site at the same time.
2- In the leave operation, barrier make sure that visitors wait for the guide.
3- In the leave operation, only the last visitor can signal the new visitors. Since current_visitor value incremented and decremented atomically (using mutex), we also ensurie there is no data racing, so the last visitor is no doubt is the last.
4- Last visitor also resets the semaphore, so that it can again be used by another tour with guide.
5- When there is no guide, we do not wait anybody, but again only the last visitor give the signal.
6- Lastly, when there is no tour, there is no blocking since a signal is given directly and atomically.

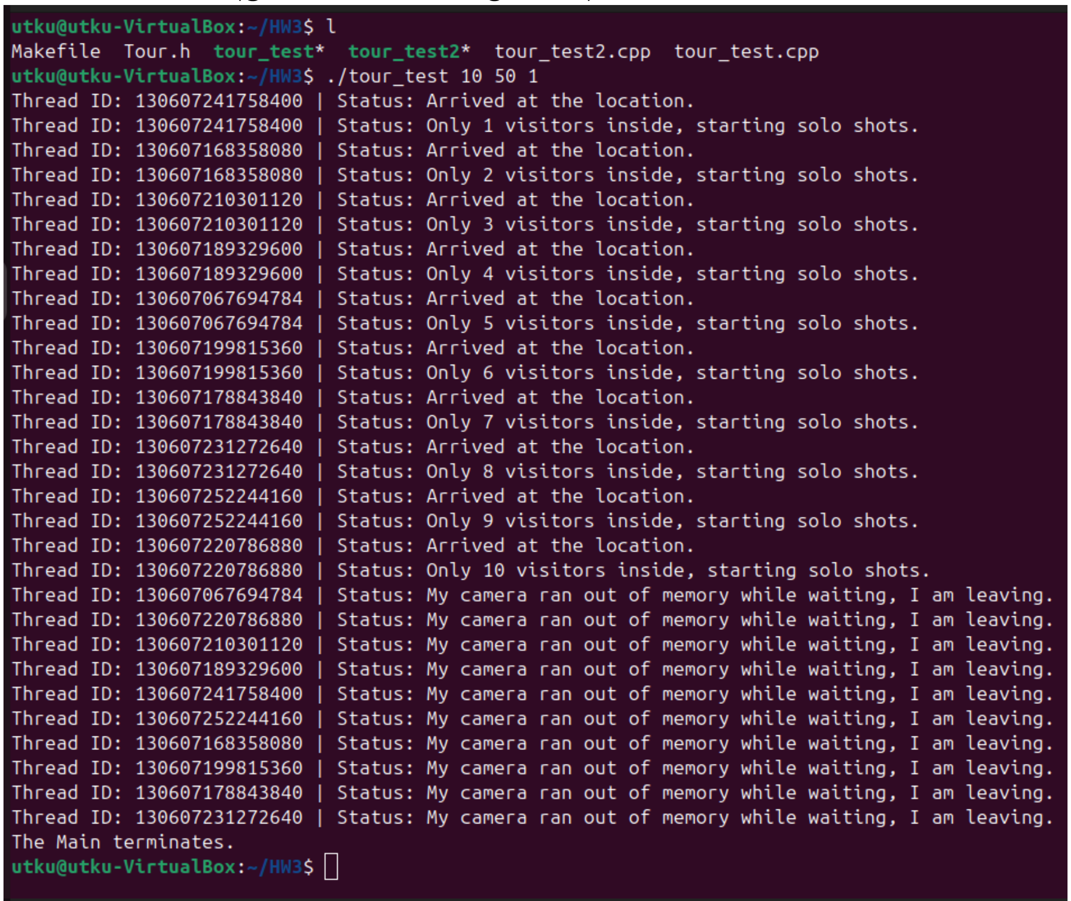
5-Samples According to Grading Criteria

Exception Handling

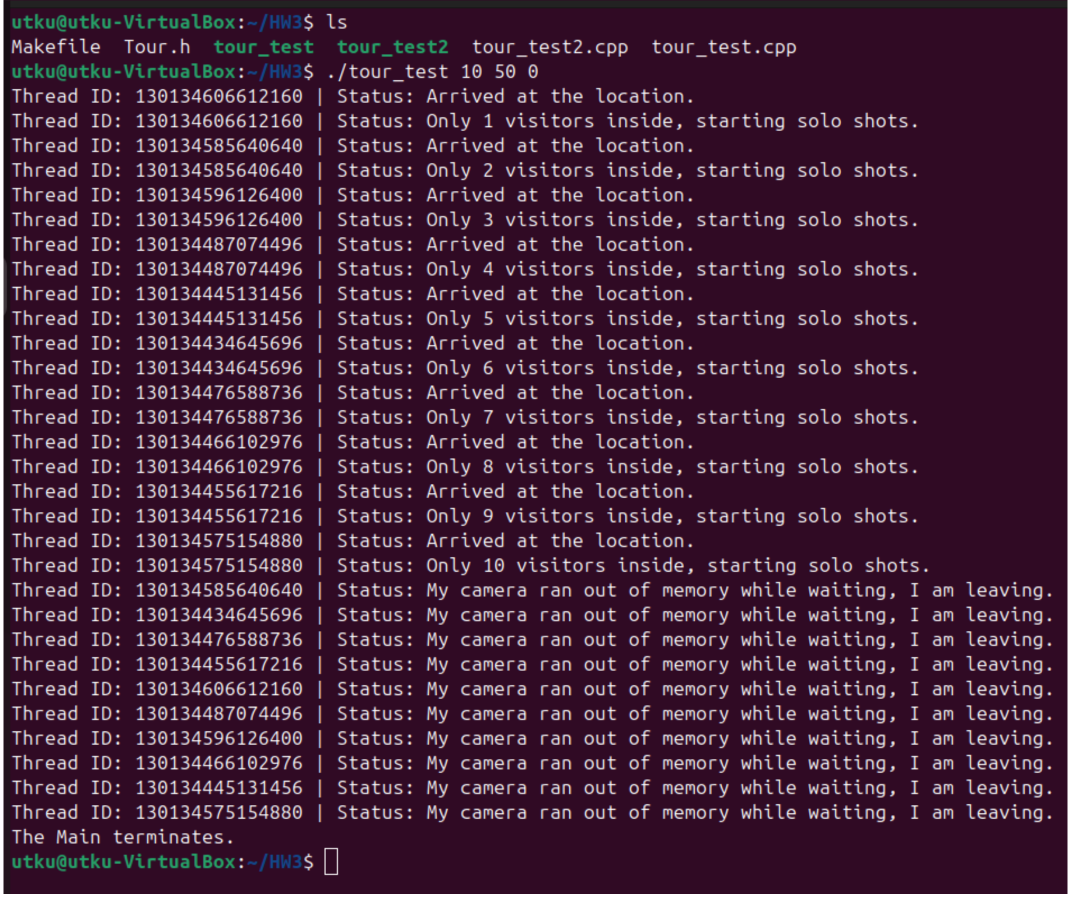


```
utku@utku-VirtualBox:~$ cd HW3
utku@utku-VirtualBox:~/HW3$ make
g++  tour_test2.cpp -o tour_test2 -lpthread
g++  tour_test.cpp -o tour_test -lpthread
utku@utku-VirtualBox:~/HW3$ ls
Makefile  Tour.h  tour_test  tour_test2  tour_test2.cpp  tour_test.cpp
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 -4 0
Exception caught:  An error occurred.
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 4 -1
Exception caught:  An error occurred.
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 4 3
Exception caught:  An error occurred.
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 0 0
Exception caught:  An error occurred.
utku@utku-VirtualBox:~/HW3$ 
```

No tours case (guided and not guided)

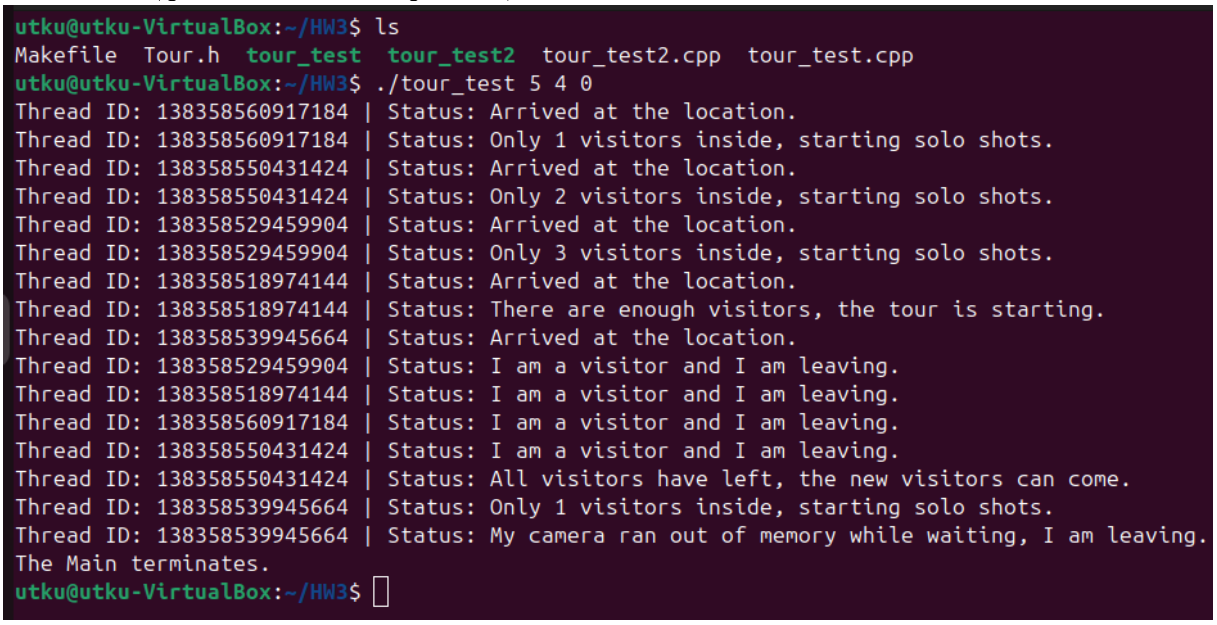


```
utku@utku-VirtualBox:~/HW3$ l
Makefile  Tour.h  tour_test*  tour_test2*  tour_test2.cpp  tour_test.cpp
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 50 1
Thread ID: 130607241758400 | Status: Arrived at the location.
Thread ID: 130607241758400 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 130607168358080 | Status: Arrived at the location.
Thread ID: 130607168358080 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 130607210301120 | Status: Arrived at the location.
Thread ID: 130607210301120 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 130607189329600 | Status: Arrived at the location.
Thread ID: 130607189329600 | Status: Only 4 visitors inside, starting solo shots.
Thread ID: 130607067694784 | Status: Arrived at the location.
Thread ID: 130607067694784 | Status: Only 5 visitors inside, starting solo shots.
Thread ID: 130607199815360 | Status: Arrived at the location.
Thread ID: 130607199815360 | Status: Only 6 visitors inside, starting solo shots.
Thread ID: 130607178843840 | Status: Arrived at the location.
Thread ID: 130607178843840 | Status: Only 7 visitors inside, starting solo shots.
Thread ID: 130607231272640 | Status: Arrived at the location.
Thread ID: 130607231272640 | Status: Only 8 visitors inside, starting solo shots.
Thread ID: 130607252244160 | Status: Arrived at the location.
Thread ID: 130607252244160 | Status: Only 9 visitors inside, starting solo shots.
Thread ID: 130607220786880 | Status: Arrived at the location.
Thread ID: 130607220786880 | Status: Only 10 visitors inside, starting solo shots.
Thread ID: 130607067694784 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607220786880 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607210301120 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607189329600 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607241758400 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607252244160 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607168358080 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607199815360 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607178843840 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130607231272640 | Status: My camera ran out of memory while waiting, I am leaving.
The Main terminates.
utku@utku-VirtualBox:~/HW3$ 
```

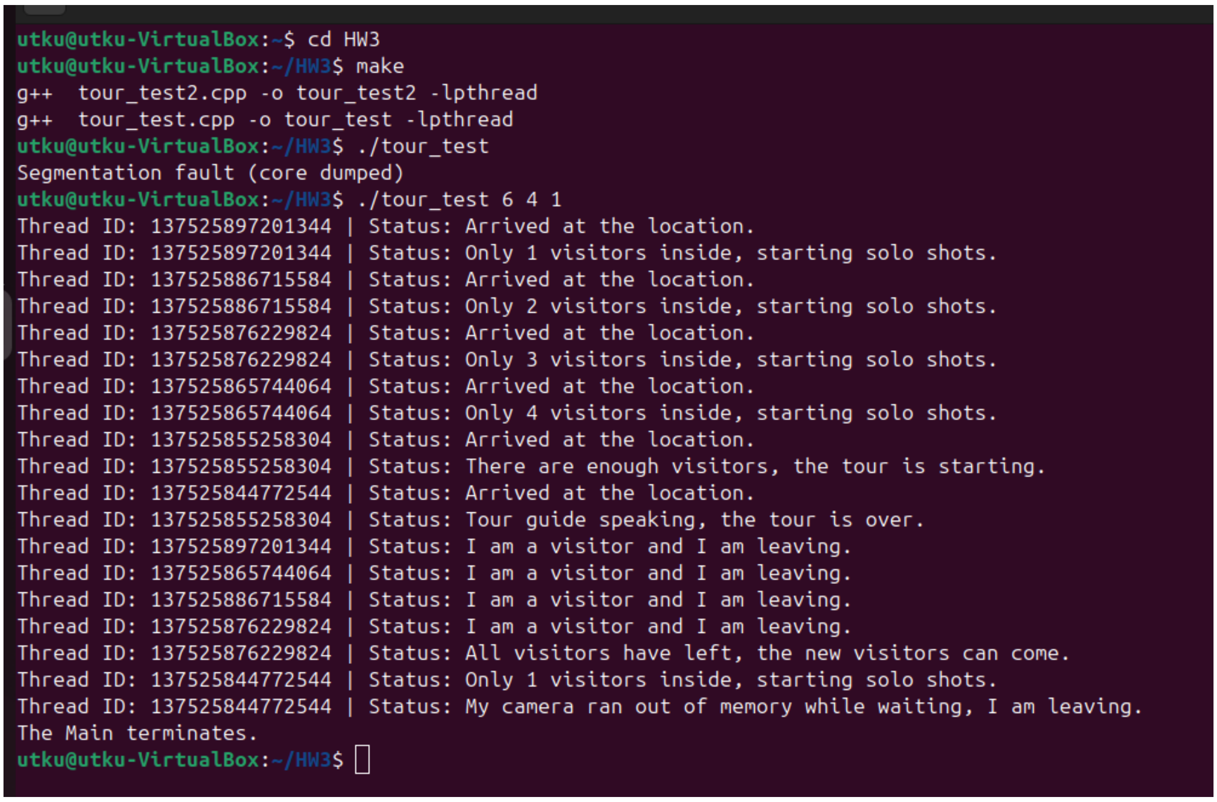


```
utku@utku-VirtualBox:~/HW3$ ls
Makefile  Tour.h  tour_test  tour_test2  tour_test2.cpp  tour_test.cpp
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 50 0
Thread ID: 130134606612160 | Status: Arrived at the location.
Thread ID: 130134606612160 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 130134585640640 | Status: Arrived at the location.
Thread ID: 130134585640640 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 130134596126400 | Status: Arrived at the location.
Thread ID: 130134596126400 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 130134487074496 | Status: Arrived at the location.
Thread ID: 130134487074496 | Status: Only 4 visitors inside, starting solo shots.
Thread ID: 130134445131456 | Status: Arrived at the location.
Thread ID: 130134445131456 | Status: Only 5 visitors inside, starting solo shots.
Thread ID: 130134434645696 | Status: Arrived at the location.
Thread ID: 130134434645696 | Status: Only 6 visitors inside, starting solo shots.
Thread ID: 130134476588736 | Status: Arrived at the location.
Thread ID: 130134476588736 | Status: Only 7 visitors inside, starting solo shots.
Thread ID: 130134466102976 | Status: Arrived at the location.
Thread ID: 130134466102976 | Status: Only 8 visitors inside, starting solo shots.
Thread ID: 130134455617216 | Status: Arrived at the location.
Thread ID: 130134455617216 | Status: Only 9 visitors inside, starting solo shots.
Thread ID: 130134575154880 | Status: Arrived at the location.
Thread ID: 130134575154880 | Status: Only 10 visitors inside, starting solo shots.
Thread ID: 130134585640640 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134434645696 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134476588736 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134455617216 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134606612160 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134487074496 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134596126400 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134466102976 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134445131456 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 130134575154880 | Status: My camera ran out of memory while waiting, I am leaving.
The Main terminates.
utku@utku-VirtualBox:~/HW3$ 
```

One tour (guided and not guided)

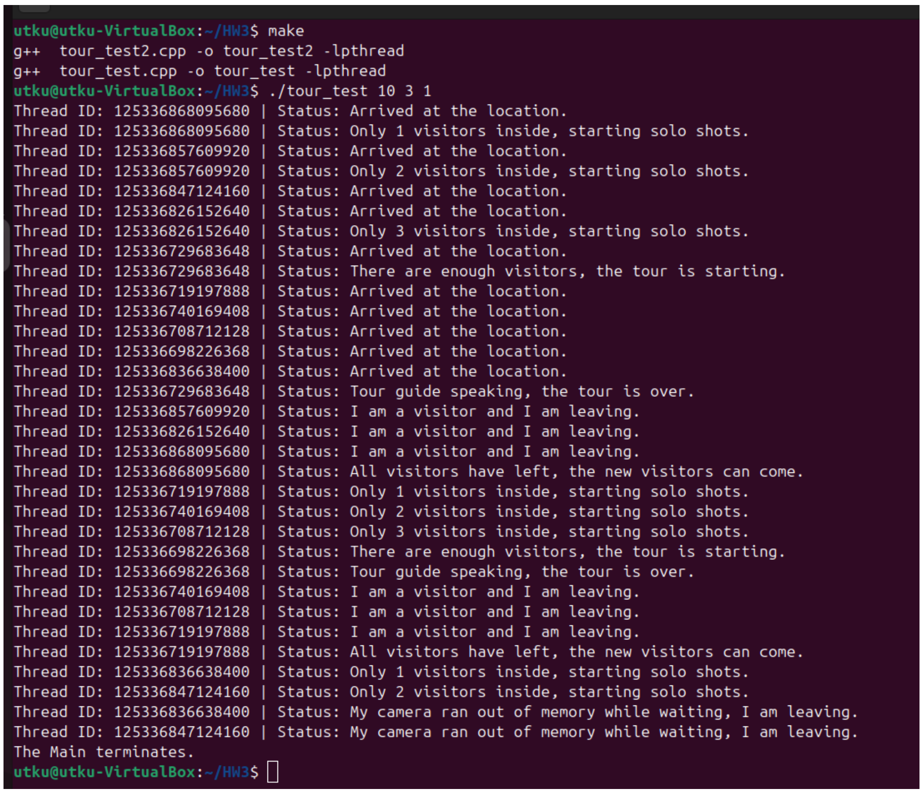


```
utku@utku-VirtualBox:~/HW3$ ls
Makefile  Tour.h  tour_test  tour_test2  tour_test2.cpp  tour_test.cpp
utku@utku-VirtualBox:~/HW3$ ./tour_test 5 4 0
Thread ID: 138358560917184 | Status: Arrived at the location.
Thread ID: 138358560917184 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 138358550431424 | Status: Arrived at the location.
Thread ID: 138358550431424 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 138358529459904 | Status: Arrived at the location.
Thread ID: 138358529459904 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 138358518974144 | Status: Arrived at the location.
Thread ID: 138358518974144 | Status: There are enough visitors, the tour is starting.
Thread ID: 138358539945664 | Status: Arrived at the location.
Thread ID: 138358529459904 | Status: I am a visitor and I am leaving.
Thread ID: 138358518974144 | Status: I am a visitor and I am leaving.
Thread ID: 138358560917184 | Status: I am a visitor and I am leaving.
Thread ID: 138358550431424 | Status: I am a visitor and I am leaving.
Thread ID: 138358550431424 | Status: All visitors have left, the new visitors can come.
Thread ID: 138358539945664 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 138358539945664 | Status: My camera ran out of memory while waiting, I am leaving.
The Main terminates.
utku@utku-VirtualBox:~/HW3$
```

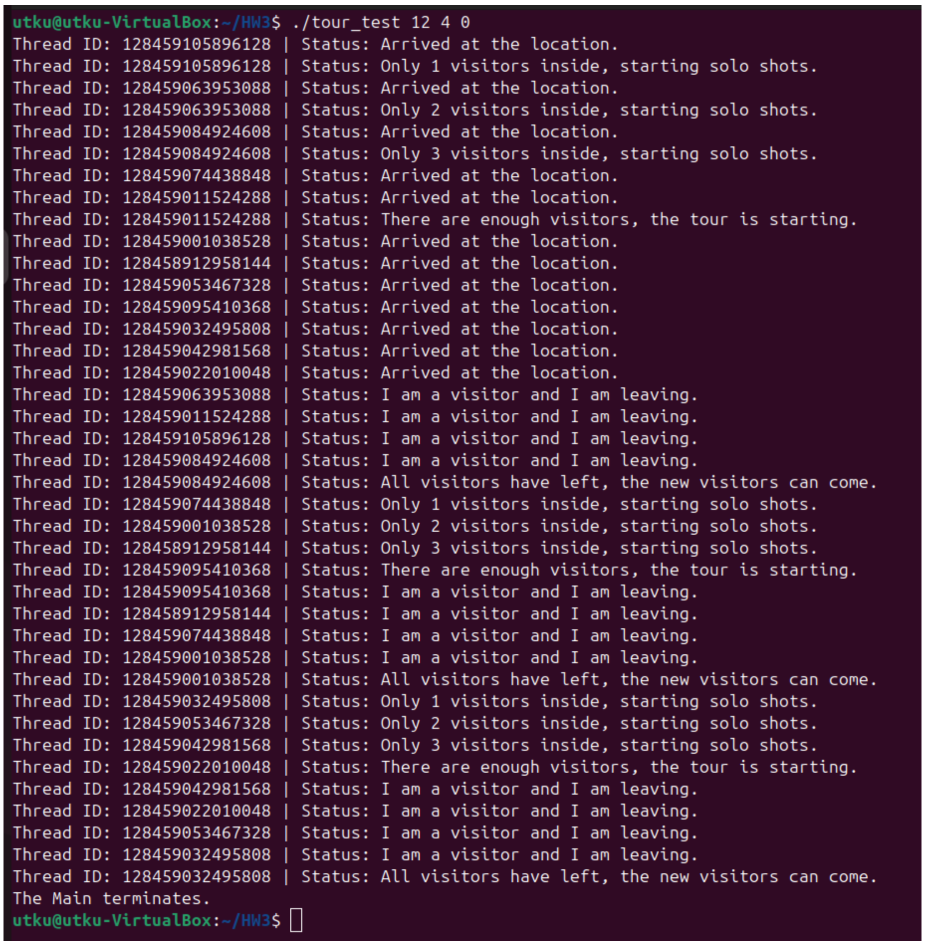


```
utku@utku-VirtualBox:~$ cd HW3
utku@utku-VirtualBox:~/HW3$ make
g++  tour_test2.cpp -o tour_test2 -lpthread
g++  tour_test.cpp -o tour_test -lpthread
utku@utku-VirtualBox:~/HW3$ ./tour_test
Segmentation fault (core dumped)
utku@utku-VirtualBox:~/HW3$ ./tour_test 6 4 1
Thread ID: 137525897201344 | Status: Arrived at the location.
Thread ID: 137525897201344 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 137525886715584 | Status: Arrived at the location.
Thread ID: 137525886715584 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 137525876229824 | Status: Arrived at the location.
Thread ID: 137525876229824 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 137525865744064 | Status: Arrived at the location.
Thread ID: 137525865744064 | Status: Only 4 visitors inside, starting solo shots.
Thread ID: 137525855258304 | Status: Arrived at the location.
Thread ID: 137525855258304 | Status: There are enough visitors, the tour is starting.
Thread ID: 137525844772544 | Status: Arrived at the location.
Thread ID: 137525855258304 | Status: Tour guide speaking, the tour is over.
Thread ID: 137525897201344 | Status: I am a visitor and I am leaving.
Thread ID: 137525865744064 | Status: I am a visitor and I am leaving.
Thread ID: 137525886715584 | Status: I am a visitor and I am leaving.
Thread ID: 137525876229824 | Status: I am a visitor and I am leaving.
Thread ID: 137525876229824 | Status: All visitors have left, the new visitors can come.
Thread ID: 137525844772544 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 137525844772544 | Status: My camera ran out of memory while waiting, I am leaving.
The Main terminates.
utku@utku-VirtualBox:~/HW3$
```

Multiple tours (guided and not guided)



```
utku@utku-VirtualBox:~/HW3$ make
g++  tour_test2.cpp -o tour_test2 -lpthread
g++  tour_test.cpp -o tour_test -lpthread
utku@utku-VirtualBox:~/HW3$ ./tour_test 10 3 1
Thread ID: 125336868095680 | Status: Arrived at the location.
Thread ID: 125336868095680 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 125336857609920 | Status: Arrived at the location.
Thread ID: 125336857609920 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 125336847124160 | Status: Arrived at the location.
Thread ID: 125336826152640 | Status: Arrived at the location.
Thread ID: 125336826152640 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 125336729683648 | Status: Arrived at the location.
Thread ID: 125336729683648 | Status: There are enough visitors, the tour is starting.
Thread ID: 125336719197888 | Status: Arrived at the location.
Thread ID: 125336740169408 | Status: Arrived at the location.
Thread ID: 125336708712128 | Status: Arrived at the location.
Thread ID: 125336698226368 | Status: Arrived at the location.
Thread ID: 125336836638400 | Status: Arrived at the location.
Thread ID: 125336729683648 | Status: Tour guide speaking, the tour is over.
Thread ID: 125336857609920 | Status: I am a visitor and I am leaving.
Thread ID: 125336826152640 | Status: I am a visitor and I am leaving.
Thread ID: 125336868095680 | Status: I am a visitor and I am leaving.
Thread ID: 125336868095680 | Status: All visitors have left, the new visitors can come.
Thread ID: 125336719197888 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 125336740169408 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 125336708712128 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 125336698226368 | Status: There are enough visitors, the tour is starting.
Thread ID: 125336698226368 | Status: Tour guide speaking, the tour is over.
Thread ID: 125336740169408 | Status: I am a visitor and I am leaving.
Thread ID: 125336708712128 | Status: I am a visitor and I am leaving.
Thread ID: 125336719197888 | Status: I am a visitor and I am leaving.
Thread ID: 125336719197888 | Status: All visitors have left, the new visitors can come.
Thread ID: 125336836638400 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 125336847124160 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 125336836638400 | Status: My camera ran out of memory while waiting, I am leaving.
Thread ID: 125336847124160 | Status: My camera ran out of memory while waiting, I am leaving.
The Main terminates.
utku@utku-VirtualBox:~/HW3$
```



```
utku@utku-VirtualBox:~/HW3$ ./tour_test 12 4 0
Thread ID: 128459105896128 | Status: Arrived at the location.
Thread ID: 128459105896128 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 128459063953088 | Status: Arrived at the location.
Thread ID: 128459063953088 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 128459084924608 | Status: Arrived at the location.
Thread ID: 128459084924608 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 128459074438848 | Status: Arrived at the location.
Thread ID: 128459011524288 | Status: Arrived at the location.
Thread ID: 128459011524288 | Status: There are enough visitors, the tour is starting.
Thread ID: 128459001038528 | Status: Arrived at the location.
Thread ID: 128458912958144 | Status: Arrived at the location.
Thread ID: 128459053467328 | Status: Arrived at the location.
Thread ID: 128459095410368 | Status: Arrived at the location.
Thread ID: 128459032495808 | Status: Arrived at the location.
Thread ID: 128459042981568 | Status: Arrived at the location.
Thread ID: 128459022010048 | Status: Arrived at the location.
Thread ID: 128459063953088 | Status: I am a visitor and I am leaving.
Thread ID: 128459011524288 | Status: I am a visitor and I am leaving.
Thread ID: 128459105896128 | Status: I am a visitor and I am leaving.
Thread ID: 128459084924608 | Status: I am a visitor and I am leaving.
Thread ID: 128459084924608 | Status: All visitors have left, the new visitors can come.
Thread ID: 128459074438848 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 128459001038528 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 128458912958144 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 128459095410368 | Status: There are enough visitors, the tour is starting.
Thread ID: 128459095410368 | Status: I am a visitor and I am leaving.
Thread ID: 128458912958144 | Status: I am a visitor and I am leaving.
Thread ID: 128459074438848 | Status: I am a visitor and I am leaving.
Thread ID: 128459001038528 | Status: I am a visitor and I am leaving.
Thread ID: 128459001038528 | Status: All visitors have left, the new visitors can come.
Thread ID: 128459032495808 | Status: Only 1 visitors inside, starting solo shots.
Thread ID: 128459053467328 | Status: Only 2 visitors inside, starting solo shots.
Thread ID: 128459042981568 | Status: Only 3 visitors inside, starting solo shots.
Thread ID: 128459022010048 | Status: There are enough visitors, the tour is starting.
Thread ID: 128459042981568 | Status: I am a visitor and I am leaving.
Thread ID: 128459022010048 | Status: I am a visitor and I am leaving.
Thread ID: 128459053467328 | Status: I am a visitor and I am leaving.
Thread ID: 128459032495808 | Status: I am a visitor and I am leaving.
Thread ID: 128459032495808 | Status: All visitors have left, the new visitors can come.
The Main terminates.
utku@utku-VirtualBox:~/HW3$
```

Thank you for reading.