East West University

Operating System

Department of CSE

Course Name: Operating system

Course Code: CSE 325

Section No: 02

Projects Name: FIFA World Cup

Submitted To.

Dr. Md Nawab Yousuf Ali
Professor, Department of
CSE, EWU

Submitted By.

1. Md Ripon AL Mamun
ID:2021-2-60-083
Department of CSE, EWU

2. Tonmoy Bhadra
ID:2020-2-60-178
Department of CSE, EWU

3. Marzan Billah Shefa
ID:2018-3-60-112

Submission Date:05-09-23

Project Title: FIFA World Cup

Project Description:

For the FIFA World Cup 2006, a fly-over has been constructed between the hotel where the teams are staying and the stadium. This fly-over will be used by the German team and the Italian team in the upcoming semifinal on Tuesday. A tram car is used to cross this fly-over, but it seats only four people, and must always carry a full load. We cannot put three Italians and one German in the same tramcar, because the Italians would be in majority and might try to intimidate the German. Similarly, we cannot put three Germans in the same tram-car with one Italian. All other combinations are safe. Implement a synchronization method to solve the problem. You must output the progress of the processes. E.g., When a player arrives you should output who arrived and also print the total number of Germans and Italians waiting to board the tram. When a tram leaves you should output that as well.

Overview:

To solve the described problem of transporting the German and Italian teams to the stadium via a tram during the FIFA World Cup 2006 semifinal match, a synchronization method using semaphores can be employed.

Problem Statement:

For the FIFA World Cup 2006, a critical transportation challenge has arisen due to the need for teams, specifically the German and Italian teams, to travel between their hotel and the stadium via a newly constructed fly-over. The issue at hand is the limited capacity of the tram car that crosses this fly-over, which can carry only four people at a time. Moreover, the tram must always carry a full load for efficiency. To ensure the safety and comfort of the teams, certain restrictions have been imposed:

Three Italians and one German cannot be placed in the same tram car, as it may lead to intimidation or discomfort for the German player.

Similarly, three Germans and one Italian cannot be placed in the same tram car.

This C program simulates the synchronization of German and Italian players boarding a tram with specific conditions, modeled after the scenario described for the FIFA World Cup 2006.

The program uses pthreads (POSIX threads) for concurrent execution and semaphores for synchronization.

- **MAX_PLAYERS**, **MAX_TRAM_CAPACITY**, **MAX_WAITING_GERMANS**, and **MAX_WAITING_ITALIANS** define the limits and conditions for player arrivals and tram capacity.

In this program,

we use three semaphores,

1. germans_sem
2. italians_sem
3. tram_sem

Two functions:

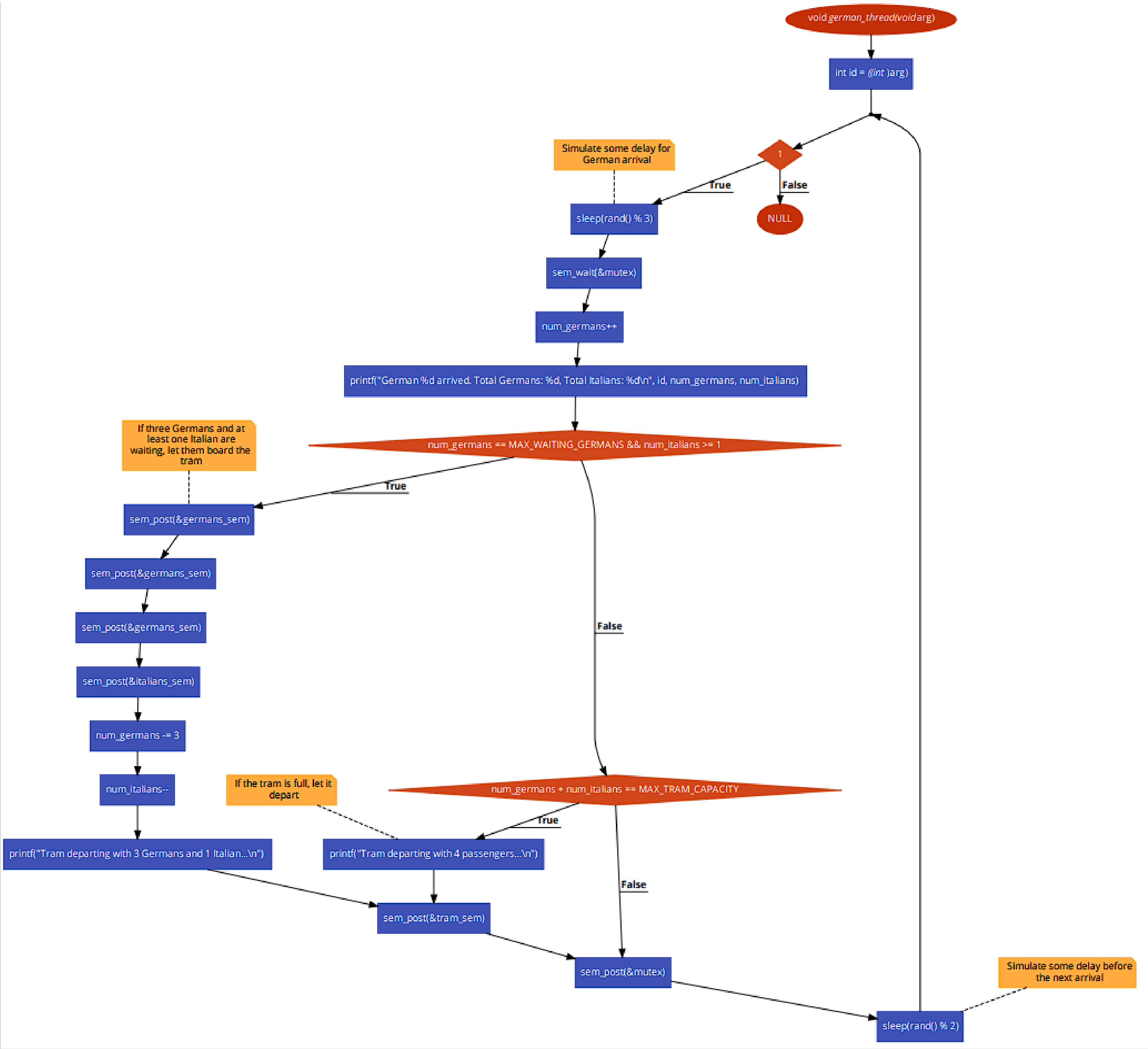
1.german_thread

2.italian_thread

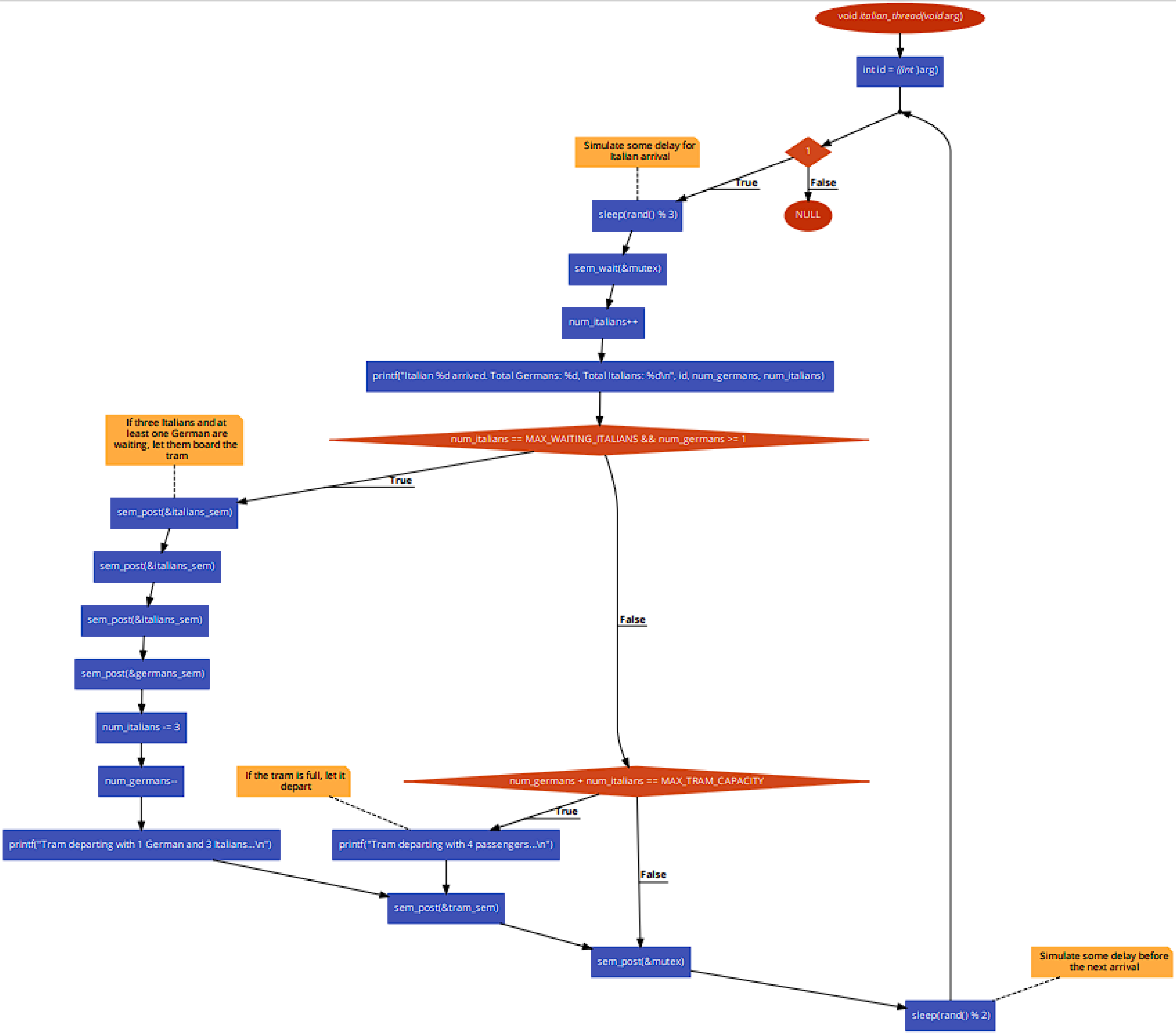
This C program simulates the synchronization of German and Italian players boarding a tram with specific conditions, modeled after the scenario described for the FIFA World Cup 2006.

Operating System: Linux

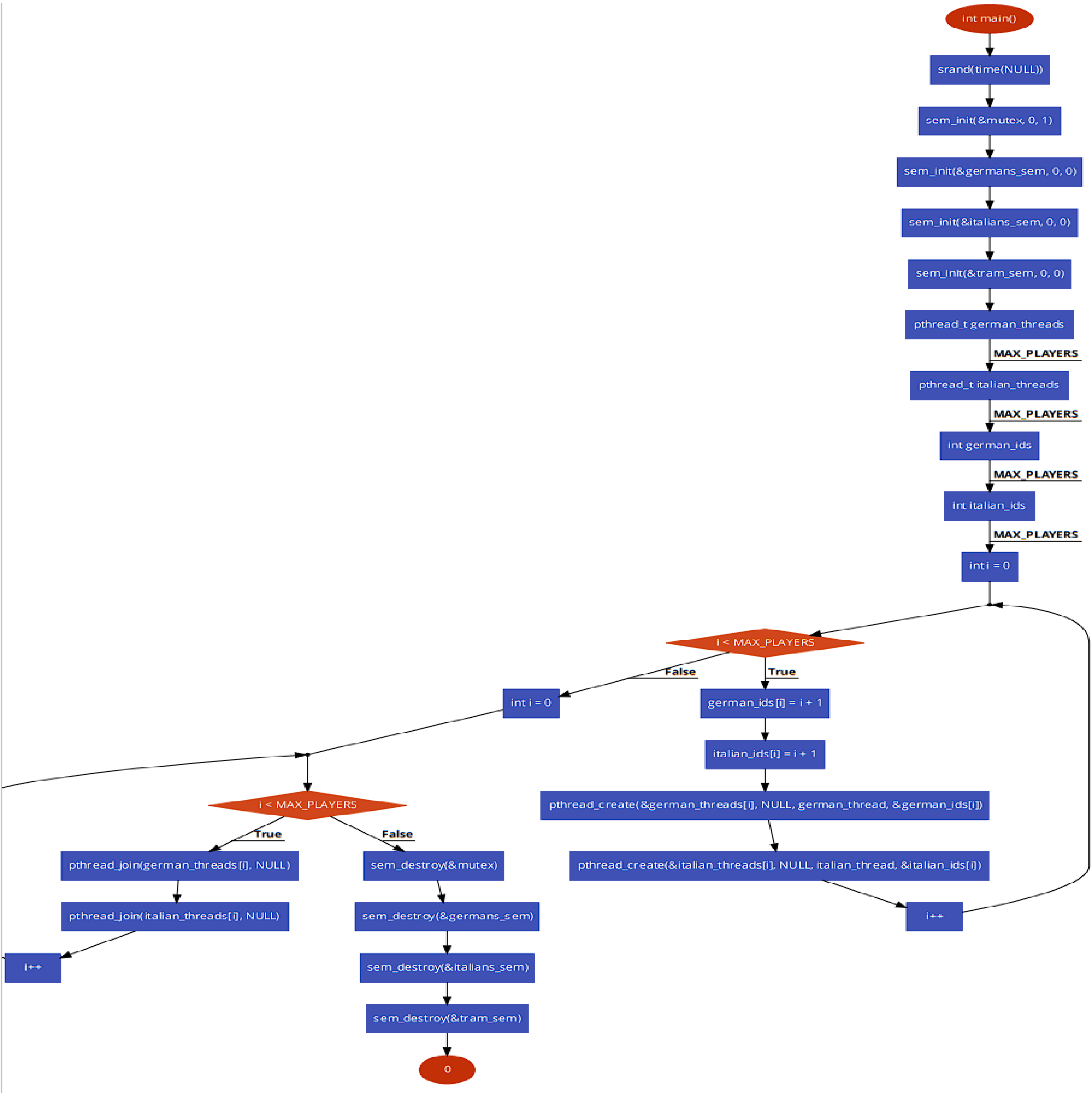
Flowchart


void german_thread(voidarg)
int id = ((int)arg)
1
True
False
NULL
Simulate some delay for German arrival
sleep(rand() % 3)
sem_wait(&mutex)
num_germans++
printf("German %d arrived. Total Germans: %d, Total Italians: %d\n", id, num_germans, num_italians)
num_germans == MAX_WAITING_GERMANS && num_italians >= 1
If three Germans and at least one Italian are waiting, let them board the tram
True
False
sem_post(&germans_sem)
sem_post(&germans_sem)
sem_post(&germans_sem)
sem_post(&italians_sem)
num_germans -= 3
num_italians--
printf("Tram departing with 3 Germans and 1 Italian...\n")
If the tram is full, let it depart
num_germans + num_italians == MAX_TRAM_CAPACITY
True
False
printf("Tram departing with 4 passengers...\n")
sem_post(&tram_sem)
sem_post(&mutex)
Simulate some delay before the next arrival
sleep(rand() % 2)

Flowchart for german_thread


void italian_thread(void arg)
int id = ((int)arg)
1
True
False
NULL
Simulate some delay for Italian arrival
sleep(rand() % 3)
sem_wait(&mutex)
num_italians++
printf("Italian %d arrived. Total Germans: %d, Total Italians: %d\n", id, num_germans, num_italians)
num_italians == MAX_WAITING_ITALIANS && num_germans >= 1
If three Italians and at least one German are waiting, let them board the tram
True
sem_post(&italians_sem)
sem_post(&italians_sem)
sem_post(&italians_sem)
sem_post(&germans_sem)
num_italians -= 3
num_germans--
printf("Tram departing with 1 German and 3 Italians...\n")
False
num_germans + num_italians == MAX_TRAM_CAPACITY
If the tram is full, let it depart
True
printf("Tram departing with 4 passengers...\n")
False
sem_post(&tram_sem)
sem_post(&mutex)
Simulate some delay before the next arrival
sleep(rand() % 2)

Flowchart for italian_thread


int main()
srand(time(NULL))
sem_init(&mutex, 0, 1)
sem_init(&germans_sem, 0, 0)
sem_init(&italians_sem, 0, 0)
sem_init(&tram_sem, 0, 0)
pthread_t german_threads
MAX_PLAYERS
pthread_t italian_threads
MAX_PLAYERS
int german_ids
MAX_PLAYERS
int italian_ids
MAX_PLAYERS
int i = 0
i < MAX_PLAYERS
False
True
int i = 0
german_ids[i] = i + 1
italian_ids[i] = i + 1
pthread_create(&german_threads[i], NULL, german_thread, &german_ids[i])
pthread_create(&italian_threads[i], NULL, italian_thread, &italian_ids[i])
i++
i < MAX_PLAYERS
True
False
pthread_join(german_threads[i], NULL)
pthread_join(italian_threads[i], NULL)
i++
sem_destroy(&mutex)
sem_destroy(&germans_sem)
sem_destroy(&italians_sem)
sem_destroy(&tram_sem)
0

Flowchart for Main function

Here is source code in c:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>

#define MAX_PLAYERS 100
#define MAX_TRAM_CAPACITY 4
#define MAX_WAITING_GERMANS 3
#define MAX_WAITING_ITALIANS 3

int num_germans = 0;
int num_italians = 0;

sem_t mutex;
sem_t germans_sem;
sem_t italians_sem;
sem_t tram_sem;

void *german_thread(void *arg) {
    int id = *((int *)arg);
    while (1) {
        // Simulate some delay for German arrival
        sleep(rand() % 3);

        sem_wait(&mutex);
        num_germans++;

        printf("German %d arrived. Total Germans: %d, Total Italians:
%d\n", id, num_germans, num_italians);

        if (num_germans == MAX_WAITING_GERMANS &&
num_italians >= 1) {
            // If three Germans and at least one Italian are waiting, let them
board the tram
            sem_post(&germans_sem);
            sem_post(&germans_sem);
            sem_post(&germans_sem);
            sem_post(&italians_sem);
            num_germans -= 3;
            num_italians--;
            printf("Tram departing with 3 Germans and 1 Italian...\n");
            sem_post(&tram_sem);
        } else if (num_germans + num_italians ==
MAX_TRAM_CAPACITY) {
            // If the tram is full, let it depart
            printf("Tram departing with 4 passengers...\n");
            sem_post(&tram_sem);
        }

        sem_post(&mutex);

        // Simulate some delay before the next arrival
        sleep(rand() % 2);
    }

    return NULL;
}

void *italian_thread(void *arg) {
    int id = *((int *)arg);

    while (1) {
        // Simulate some delay for Italian arrival
        sleep(rand() % 3);

        sem_wait(&mutex);
        num_italians++;
        printf("Italian %d arrived. Total Germans: %d, Total Italians:
%d\n", id, num_germans, num_italians);

        if (num_italians == MAX_WAITING_ITALIANS &&
num_germans >= 1) {
            // If three Italians and at least one German are waiting, let them
board the tram
            sem_post(&italians_sem);
            sem_post(&italians_sem);
            sem_post(&italians_sem);
            sem_post(&germans_sem);
            num_italians -= 3;
            num_germans--;
            printf("Tram departing with 1 German and 3 Italians...\n");
            sem_post(&tram_sem);
        } else if (num_germans + num_italians ==
MAX_TRAM_CAPACITY) {
            // If the tram is full, let it depart
            printf("Tram departing with 4 passengers...\n");
            sem_post(&tram_sem);
        }

        sem_post(&mutex);

        // Simulate some delay before the next arrival
```

```
        sleep(rand() % 2);
    }

    return NULL;
}

int main() {
    srand(time(NULL));

    sem_init(&mutex, 0, 1);
    sem_init(&germans_sem, 0, 0);
    sem_init(&italians_sem, 0, 0);
    sem_init(&tram_sem, 0, 0);

    pthread_t german_threads[MAX_PLAYERS];
    pthread_t italian_threads[MAX_PLAYERS];

    int german_ids[MAX_PLAYERS];
    int italian_ids[MAX_PLAYERS];

    for (int i = 0; i < MAX_PLAYERS; i++) {
        german_ids[i] = i + 1;
        italian_ids[i] = i + 1;
        pthread_create(&german_threads[i], NULL, german_thread,
&german_ids[i]);
        pthread_create(&italian_threads[i], NULL, italian_thread,
&italian_ids[i]);
    }

    for (int i = 0; i < MAX_PLAYERS; i++) {
        pthread_join(german_threads[i], NULL);
        pthread_join(italian_threads[i], NULL);
    }

    sem_destroy(&mutex);
    sem_destroy(&germans_sem);
    sem_destroy(&italians_sem);
    sem_destroy(&tram_sem);

    return 0;
}
```