**Q2:**

Semaphore full = 0; // Semaphore to signal when the stack is full

Semaphore empty = MAX; // Semaphore to signal when the stack is empty

Semaphore mutex = 1; // Semaphore to ensure mutual exclusion for stack operations

class stack {

Private:

int \*arr;

int max;

int top;

public:

stack(int n) {

arr = new int[n];

max = n;

top = 0;

}

void push(int x) {

wait(empty);

wait(mutex);

arr[top] = x;

++top;

signal(mutex);

signal(full);

}

int pop() {

wait(full);

wait(mutex);

int temp = top;

–top;

signal(mutex);

signal(empty);

return arr[temp];

}

};

**Q3:**

semaphore entry\_mutex = 1;

semaphore drama\_enthusiast \_mutex = 1;

semaphore comedy\_fan\_mutex = 1;

int drama\_enthusiast \_count = 0;

int comedy\_fan\_count = 0;

**1. comedy\_fan \_wants\_to\_enter**

wait (comedy\_fan\_mutex);

comedy\_fan\_count++;

if (comedy\_fan\_count == 1) wait(entry\_mutex);

signal(comedy\_fan\_mutex);

[comedy show in progress]

**2. comedy\_fan\_leaves**

wait(comedy\_fan\_mutex);

comedy\_fan\_count--;

if (comedy\_fan\_count == 0) signal (entry\_mutex);

signal(comedy\_fan\_mutex);

**3. drama enthusiast wants to enter**

wait (drama\_enthusiast\_mutex);

drama\_enthusiast \_count++;

if (drama\_enthusiast \_count == 1) wait(entry\_mutex);

signal(drama\_enthusiast \_mutex);

[drama show in progress]

**4. drama\_enthusiast\_leaves**

wait(drama\_enthusiast \_mutex);

drama\_enthusiast \_count--;

if (drama\_enthusiast \_count == 0) signal (entry\_mutex);

signal(drama\_enthusiast \_mutex);