Q. Consider the following concurrently running processes P0 and P1.

Shared data:

bool *waiting [0] = false, waiting [1] = false;*
int *turn = 0;*

Process P0:

```
while (true)
{
a1      waiting [0] = true;
a2      while (turn != 0)
        {
a3              while (waiting [1]);
a4              turn = 0;
        }
        Critical Section code
        waiting [0] = false;
        Remainder Section code
}
```

Process P1:

```
while (true)
{
b1      waiting [1] = true;
b2      while (turn != 1)
        {
b3              while (waiting [0]);
b4              turn = 1;
        }
        Critical Section code
        waiting [1] = false;
        Remainder Section code
```

Give a sequence of instructions in terms of **ai**'s and **bi**'s to show that mutual exclusion is not satisfied.

Sequence:

Q.3. (15 points) There are 3 robots (red, blue, green) – each is controlled by its own process. We need to ensure that the robots only move in the following order: red, blue, green, red, blue, green, etc. Add the necessary code below that performs the appropriate initializations and enforces this execution order. Use only semaphores for your synchronization.

Shared Variables:

Semaphore .....R=1....., ..B=0....., G=0.....;

```
Process Robot_red
do {

    MOVE();

}while(true);
```

```
Process Robot_blue
do {

    MOVE();

}while(true);
```

```
Process Robot_green
do {

    MOVE();

}while(true);
```