```
        }
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0;
        while (i < 05) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i = 0;
        while (i < 05) {
            int r = q.get();
            System.out.println("Consumed: " + r);
```

Lab 10

1) Demonstrate Inter process communication and deadlock

Pgm a : To demonstrate Inter process comm"

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while (!valueSet)
            try {
                System.out.println("In consumer waiting
                                    \n");
                wait();
            }
            catch (InterruptedException e)
            {
                System.out.println("In Interrupted exception
                caught");
            }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("In Intimate Producer \n");
        notify();
        return n;
    }
    Synchronized void put(int n) {
        while (valueSet)
            try
            {
```

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("A interrupted");
        }
        System.out.println(name + " trying to call B.last()");
        b.last();
    }
    void last() {
        System.out.println("Inside A.last");
    }
}

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("Interrupted");
        }
    }
```

Consumed : 1

Pw : 2

Initiate Consumer

Producer waiting

Cns : 2

Initiate producer

Consumed : 2

Pw : 3

Initiate ~~Producer~~ Consumer

Produce waiting

Cns : 3

Initiate Producer

Consumed : 3

Pw : 4

Initiate Consumer

Cns : 4

Initiate Producer

Consumed : 4

Pgm

Clas

Date
13-2-24

```
}
}
class PCFixed {
public static void main (String args[]) {
    Q q = new Q();
    new Producer (q);
    new Consumer (q);
    System.out.println ("Press Control-c
    to stop.");
}
}
```

output

Press Control-c to stop.

Put: 0

Initiate Consumer

Producer waiting

Got: 0

Initiate Producer

Put: 1

Initiate Consumer

Producer waiting

Consumed: 0

Got: 1

Initiate Producer

Report

Main Thread entered A..foo
Racing Thread entered B.bar

Racing Thread ~~entered~~ trying to call A.last().

Inside A.last
Back in other Thread

main Thread trying to call B.last()
Inside A.last

Back in main thread.

Dec 11-2-24

```
        System.out.println(name + "Trying to
        call A.last");
        a.last();
    }
    void last() {
        System.out.println("Inside A.last");
    }
}

class Deadlock implements Runnable
{
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main thread");
    }
    public void run() {
        b.bar(a);
        System.out.println("Back in other thread");
    }
    public static void main(String args[]) {
        new Deadlock();
    }
}
```