PROGRAM 10:

Demonstrate Inter process Communication

10 A)

CODE:

```
class Q {
    int n;
    boolean valueSet = false;

    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\nNotify Producer\n");
        notify();
        return n;
    }

    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        this.n = n;
```

```
        valueSet = true;
        System.out.println("Put: " + n);
        System.out.println("\nNotify Consumer\n");
        notify();
    }
}

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 < 15) {
            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 < 15) {
            int r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

public 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:

```
Consumed: 1
Put: 2

Notify Consumer


Producer waiting

Got: 2

Notify Producer

Consumed: 2
Put: 3

Notify Consumer


Producer waiting

Got: 3

Notify Producer

Consumed: 3
Put: 4

Notify Consumer


Producer waiting
```

```
Got: 4

Notify Producer

Consumed: 4
Put: 5

Notify Consumer


Producer waiting

Got: 5

Notify Producer

Consumed: 5
Put: 6

Notify Consumer


Producer waiting

Got: 6

Notify Producer

Consumed: 6
Put: 7

Notify Consumer


Producer waiting

Got: 7

Notify Producer
```

```
Consumed: 7
Put: 8

Notify Consumer


Producer waiting

Got: 8

Notify Producer

Consumed: 8
Put: 9

Notify Consumer


Producer waiting

Got: 9

Notify Producer

Consumed: 9
Put: 10

Notify Consumer


Producer waiting

Got: 10

Notify Producer

Consumed: 10
Put: 11
```

```
Notify Consumer


Producer waiting

Got: 12

Notify Producer

Consumed: 12
Put: 13

Notify Consumer


Producer waiting

Got: 13

Notify Producer

Consumed: 13
Put: 14

Notify Consumer

Got: 14

Notify Producer

Consumed: 14
```

NOTES:

Lab - 10

PAGE NO :
DATE :

a) Demonstrate Interprocess communication and deadlock.

=>

```
class Q {
 int n;
 boolean valueset = false;
 Synchronized int get() {
 while(!valueset)
 try {
 System.out.println("\n customer waiting");
 wait();
 }
 catch (InterruptedException e) {
   Sout("Interrupted Exception caught");
 }
 Sout("Got " + n);
 valueset = false;
 Sout("Intimate producer");
 notify();
 return n;
 }
 synchronized void put(int n) {
 while(valueset)
 try {
 Sout("producer waiting");
 wait();
 }
 catch (InterruptedException e)
 {
 Sout("Interrupted Exception caught");
 }
 this.n = n;
```

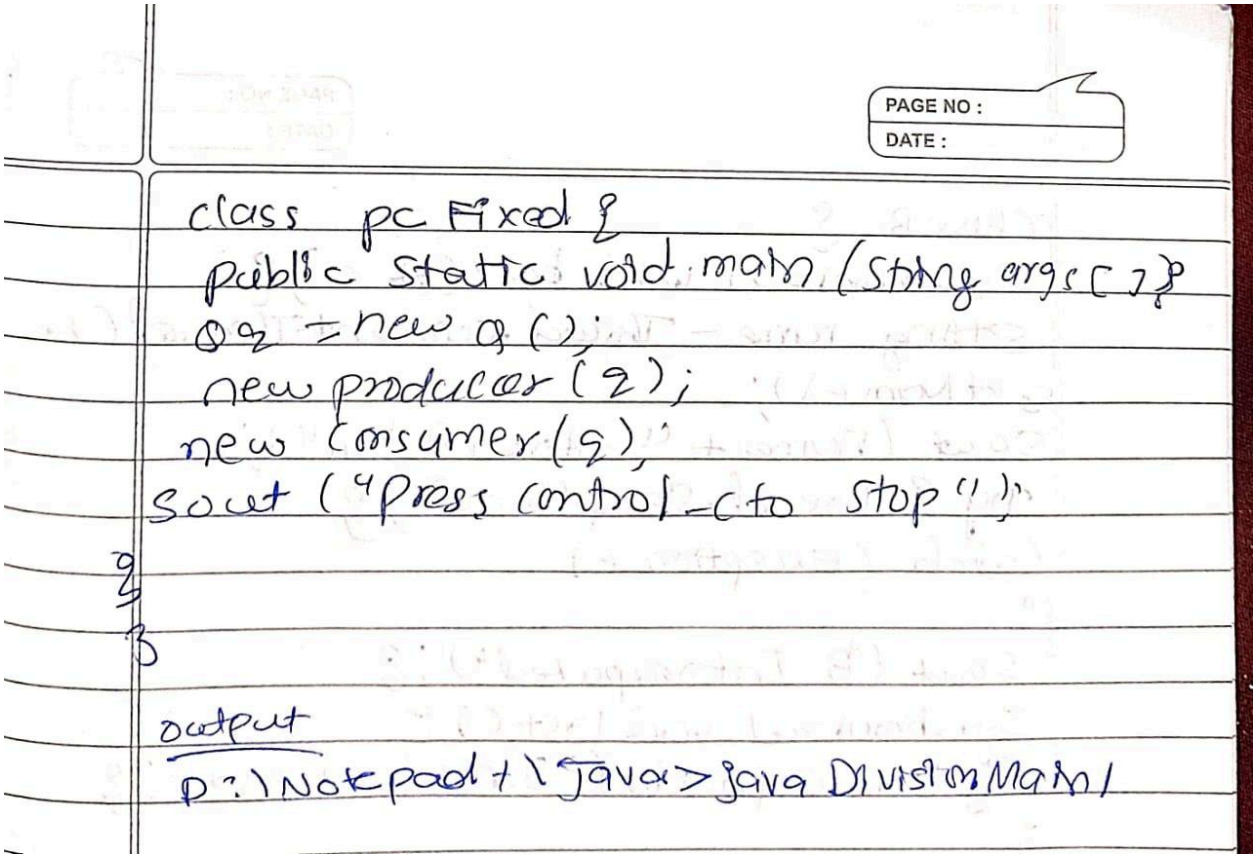
PAGE NO :
DATE :

```
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<15){
 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<15){
int r= q.get();
Sout ("consumed "+r);
 i++;
}
}
}
```



10 B)
DEMONSTRATION OF DEADLOCK

CODE:

```
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();
    }
```

```
    synchronized 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("B Interrupted");
        }
        System.out.println(name + " trying to call A.last()");
        a.last();
    }

    synchronized void last() {
        System.out.println("Inside B.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); // get lock on a in this thread.
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // get lock on b in other thread.
        System.out.println("Back in other thread");
    }
```

```
    public static void main(String args[]) {
        new Deadlock();
    }
}
```

OUTPUT:

```
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
```

NOTES:

(5) Demonstration of deadlock.

```
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();}
Synchronized void last(){
System.out.println("Inside A.last"); }
}
```

PAGE NO :
DATE :

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

class Deadlock Implementation Runable
{
A a = new A();
  B b = new B();

Deadlock() {
Thread.currentThread().setName("
Main Thread");
Thread t = new Thread (this, "Racing
Thread");
t.start();
a.foo(b);
Sout ("Back in main Thread");
}
public void run() {
b.bar(a);
Sout ("Back in other thread");
}
public static void main (String args[])
{ new Deadlock(); }
}
```