PAGE NO :
DATE :

~~Count~~ contd Lab Program 10.

```
class Q
{
int n;
boolean valueset=false;
synchronized int get()
{
  while(!valueset)
try{
  System.out.println("Consumer waiting");
 wait();
}
catch(InterruptedException e)
{
  System.out.println("InterruptedException
                      caught");
 }
System.out.println("got:" +n);
valueSet=false;
System.out.println("Intimate Producer");
notify();
return n;
}
synchronized void put(int n)
{
 while(valueset)
try{
System.out.println("Producer waiting");
wait();
}
catch(InterruptedException e)
```

DATE :

```
{
System.out.println("Interrupted Exception
    caught");
}
this.n=n;
valueset=true;
System.out.println("Put: " +n);
System.out.println("Intimate consumer");
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++;
}
}
}
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
Terminate consumer
Producer waiting
```

got: 0
Intimate Producer
Put: 1
Intimate consumer
Producer waiting
Consumed: 0
got: 1.
Intimate Producer.
Consumed: 1
Put: 2
Intimate Consumer
Producer waiting
got: 2.
Intimate Producer
consumed: 2.
Put: 3.
Intimate consumer
Producer waiting
got: 3.
Intimate Producer
~~Put: 4. consumed: 3~~
Put: 4
Intimate consumer
Producer waiting
got: 4
Intimate Producer
Consumed: 4
Put: 5
Intimate Consumer
Producer waiting
got: 5
Intimate Producer consumed: 5

PAGE NO :
DATE :

10b) Deadlocks

```
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("B Interrupted");
    }
    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();
}
}
```

Output

```
Main thread entered A.foo
Racing thread entered B.bar.
main thread trying to call B.last()
Inside A.last
Racing Thread trying to call A.last()
Back in main thread.
Inside A.last
Back in other thread.
```

8
13/2 bm