PAGE NO :
DATE :

10.a) Demonstrate inter process communication and deadlock.

```
Class Q {
int n;
boolean valueSet = false;
Synchronized int get() {
 while(!valueSet)
 try {
 System.out.println("\n Consumer Waiting\n");
 wait();
 } catch(InterruptedException e)
 {
 System.out.println("\nInterruptedException caught");
 }
 System.out.println("Got: "+n);
 valueSet = false;
 System.out.println("\n Intimate Producer\n");
 notify();
 return n;
 }
 Synchronized void put(int n) {
 while(valueSet)
 try {
 System.out.println("\n Producer waiting\n");
 wait();
 } catch(InterruptedException e)
 {
 System.out.println("InterruptedException caught");
 }
 this.n = n;
 ValueSet = true;
 System.out.println("Put: " +n);
```

```
System.out.println("In IntStack Consumer In");
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<6){
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<6){
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:

Put: 0
Intimate Consumer
Producer Waiting
Press Control-C to Stop
Got: 0
Intimate Producer
Consumed: 0
Put: 1
Intimate Consumer
Producer Waiting
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
Consumed: 3
Put: 4
Intimate Consumer
Producer waiting
Cnt: 4
Intimate producer
Consumed: 4
Put: 5
Intimate Consumer
~~Producer Waiting~~
Cnt: 5
Intimate Producer
Consumed: 5

Shalhandi Patel C.J., IBM22CS255

b)

```
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("Main Thread");
  Thread t = new Thread(this, "Racing Thread");
  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
Back in main thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread

Shashank Patil C.J. 1BM22CS255