Manoj M Mallya
200905130
Section : C2
Roll no : 23

LAB – 6

EXERCISE QUESTIONS :

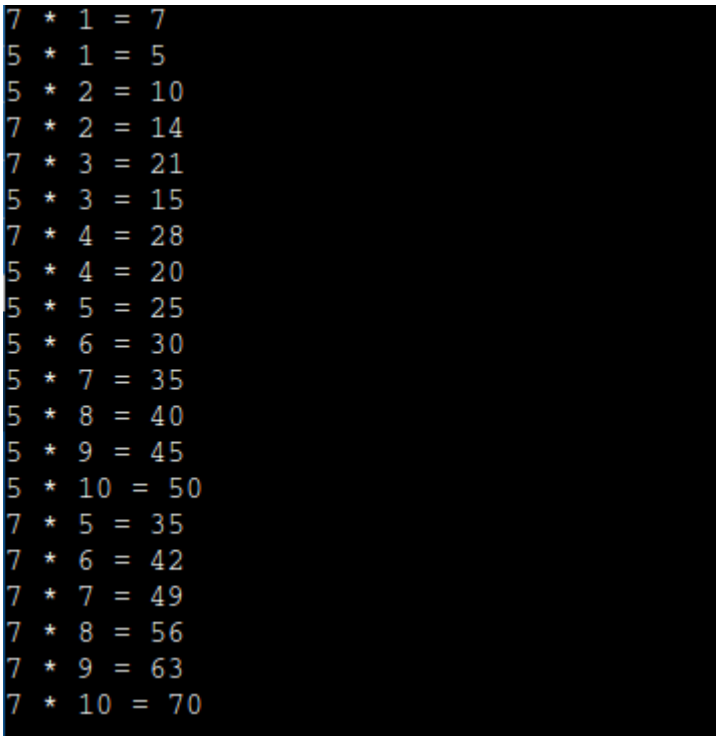
1. Create a class MxNTableThread by extending Thread class. The thread calls a non-static printTable method of another class to display multiplication table of a number supplied as parameter. Create another class TablesDemo which will instantiate two objects of the MxNTableThread class to print multiplication table of 5 and 7. Observe intermixed output from the 2 threads. Also, observe output by applying synchronization concept.

CODE :

```
class P{
void PrintTable(int x){
for(int i=1;i<11;i++)
System.out.println(x+" * "+i+" = "+i*x);
}
}

class MxNTableThread extends Thread{
int count;
P table=new P();
MxNTableThread(String name,int x){
super(name);
count=x;
start();
}

public void run(){
table.PrintTable(count);
}

}

public class TableDemo{
public static void main(String[] args) {
```

```
MxNTableThread m5=new MxNTableThread("Table_of_5",5);
MxNTableThread m7=new MxNTableThread("Table_of_7",7);
}
}
```

OUTPUT :



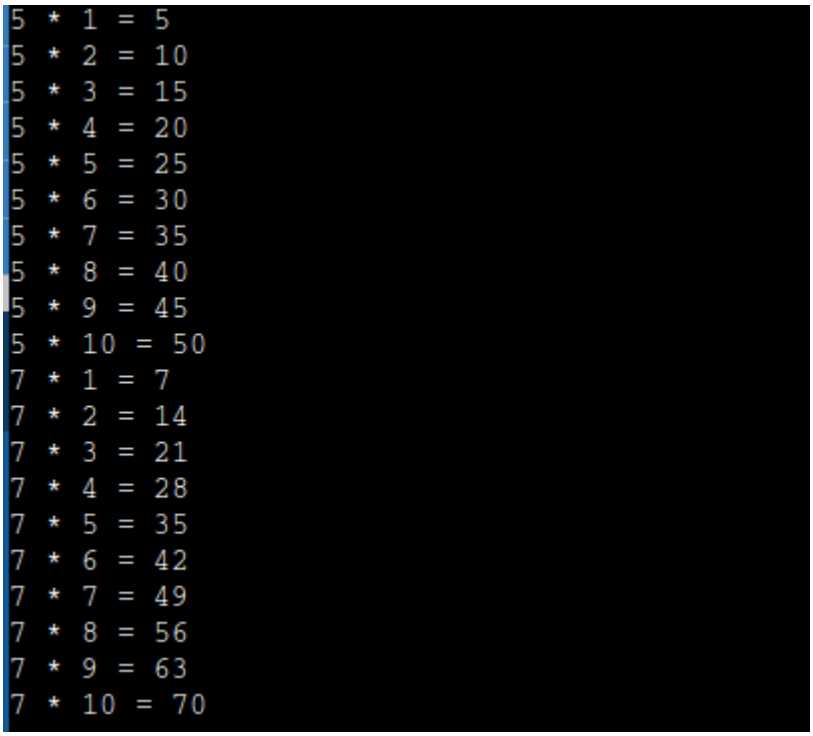
After applying synchronization,

CODE :

```
class P{
void PrintTable(int x){
for(int i=1;i<11;i++)
System.out.println(x+" * "+i+" = "+i*x);
}
}
class MxNTableThread extends Thread{
int count;
P table=new P();
MxNTableThread(String name,int x){
super(name);
count=x;
start();}
```

```
synchronized public void run(){
table.PrintTable(count);
}
}
public class TableDemo{
public static void main(String[] args) {
MxNTableThread m5=new MxNTableThread("Table_of_5",5);
MxNTableThread m7=new MxNTableThread("Table_of_7",7);
}
}
```

OUTPUT :



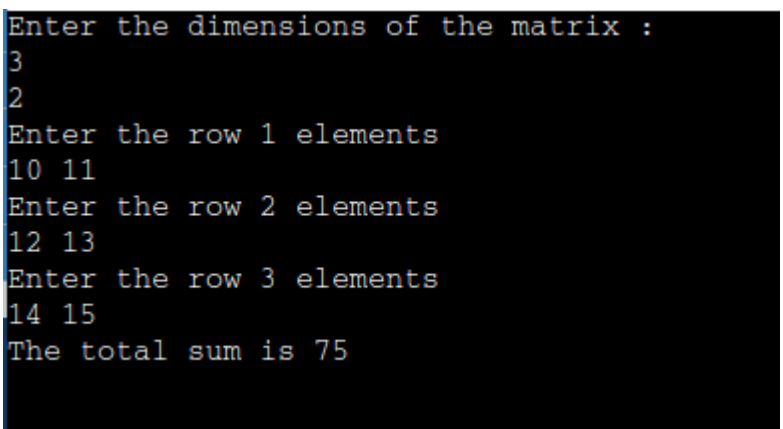
2. Write and execute a java program to create and initialize a matrix of integers. Create n threads(by implementing Runnable interface) where n is equal to the number of rows in the matrix. Each of these threads should compute a distinct row sum. The main thread computes the complete sum by looking into the partial sums given by the threads. Use join method to ensure that the main thread terminates last.

CODE :

```
 import java.util.Scanner;
class rowsum implements Runnable{
Thread thrd;
int arr[],sum=0;rowsum(String name,int arr[]){
```

```
thrd =new Thread(this,name);
thrd.start();
this.arr=arr;
}
public void run(){
for(int i=0;i<arr.length;i++)
sum+=arr[i];
}
}
public class matrixsum{
public static void main(String args[]){
System.out.println("Enter the dimensions of the matrix : ");
Scanner sc=new Scanner(System.in);
int x=sc.nextInt();
int y=sc.nextInt();
int total=0;
rowsum r[]=new rowsum[x];
int arr[][]=new int[x][y];
for(int i=0;i<x;i++){
System.out.println("Enter the row "+(i+1)+" elements");
for(int j=0;j<y;j++)
arr[i][j]=sc.nextInt();
}
for(int i=0;i<x;i++)
r[i]=new rowsum("Thread "+(i+1),arr[i]);
for(int i=0;i<x;i++)
try{
r[i].thrd.join();
}
catch(InterruptedException exc){
System.out.println("Thread interrupted");
}
for(int i=0;i<x;i++)
total+=r[i].sum;
System.out.println("The total sum is "+total);
}
}
```

OUTPUT :



3. Write and execute a java program to implement a producer - consumer problem using Inter-thread communication.

CODE :

```
class Q{
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
wait();
}
catch (InterruptedException e){
System.out.println("InterruptedException caught");
}
System.out.println("Consumer consumed : " + n);
valueSet = false;
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
wait();
}
catch (InterruptedException e){
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Producer produced : " + 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(true) {
q.put(i++);
```

```
}
}
}
class Consumer implements Runnable{
Q q;
Consumer(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
while(true) {
q.get();
}
}
}
public class prodcon{
public static void main(String[] args) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Ctrl+C to stop...");
}
}
```

OUTPUT :

```
Student@prg19:~/Desktop/200905130/lab6$ java prodcon
Press Ctrl+C to stop...
Producer produced : 0
Consumer consumed : 0
Producer produced : 1
Consumer consumed : 1
Producer produced : 2
Consumer consumed : 2
Producer produced : 3
Consumer consumed : 3
Producer produced : 4
Consumer consumed : 4
Producer produced : 5
Consumer consumed : 5
Producer produced : 6
Consumer consumed : 6
Producer produced : 7
Consumer consumed : 7
Producer produced : 8
Consumer consumed : 8
Producer produced : 9
Consumer consumed : 9
Producer produced : 10
Consumer consumed : 10
```

Result terminates after pressing ctrl+c...

****