ASSIGNMENT 12

1) Write a java program to create two threads. First thread should find the square of the number, second thread should find the sum of the digits of the squared number.

```java
import java.util.Scanner;

class Global {
    public static int num;

    public static void update(int n) {
        num = n;
    }
}

class Child1 extends Thread {
    Child1() {
        start();
    }

    public void run() {
        int n = Global.num;
        Global.update(n * n);
    }
}

class Child2 extends Thread {
    Child2() {
        start();
    }

    public void run() {
        int n = Global.num;
        int r, sum = 0;
        while (n > 0) {
            r = n % 10;
            n = n / 10;
            sum += r;
        }
        Global.update(sum);
    }
}

class GlobalDemo {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number:");
        int n = sc.nextInt();
        System.out.print("Sum of the digits of square of " + n + "=");
        Global.update(n);
        Child1 c1 = new Child1();
        Child2 c2 = new Child2();
        try {
            c1.join();
            c2.join();
        } catch (InterruptedException e) {
            System.out.print(e);
        }
        System.out.println(Global.num);
    }
}
```

OUTPUT:

```
Enter a number:
5
Sum of the digits of square of 5=7
```

2) Write a java program that will compute product of two vector (1D array) using multithreading. The program should read two vectors (of same size) from the user. First thread should multiply the corresponding elements present in the odd index position and second thread should multiply the corresponding elements present in the even index position. Main thread should display the result.

```
import java.util.Scanner;

class Global {
    public static int R[];

    Global(int s) {
        R = new int[s];
        for (int i = 0; i < s; i++)
            R[i] = 0;
    }

    public static void update(int index, int n) {
        R[index] = n;
    }
}

class Child1 extends Thread {
    int V1[], V2[];
    int size;

    Child1(int V1[], int V2[], int size) {
        this.V1 = V1;
        this.V2 = V2;
        this.size = size;
        start();
    }

    public void run() {
        try {
            for (int i = 1; i < size; i += 2) {
                Global.update(i, V1[i] * V2[i]);
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Child 1 interrupted.");
        }
    }
}

class Child2 extends Thread {
    int V1[], V2[];
```

```java
    int size;

    Child2(int V1[], int V2[], int size) {
        this.V1 = V1;
        this.V2 = V2;
        this.size = size;
        start();
    }

    public void run() {
        try {
            for (int i = 0; i < size; i += 2) {
                Global.update(i, V1[i] * V2[i]);
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Child 2 interrupted.");
        }
    }
}

class VectorDemo {
    public static void main(String args[]) {
        int size;
        int m[], n[];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the vectors:");
        size = sc.nextInt();
        m = new int[size];
        n = new int[size];
        new Global(size);
        System.out.println("Enter the first vector:");
        for (int i = 0; i < size; i++)
            m[i] = sc.nextInt();
        System.out.println("Enter the second vector:");
        for (int i = 0; i < size; i++)
            n[i] = sc.nextInt();
        Child1 c1 = new Child1(m, n, 5);
        Child2 c2 = new Child2(m, n, 5);
        try {
            c1.join();
            c2.join();
        } catch (InterruptedException e) {
            System.out.print(e);
        }
        System.out.println("Product of the vectors:");
        for (int i = 0; i < size; i++) {
            System.out.print(Global.R[i] + "\t");
        }
    }
}
```

OUTPUT:

```
Enter the size of the vectors:
5
Enter the first vector:
1 2 3 4 5
Enter the second vector:
5 4 3 2 1
```

```
Product of the vectors:
5    8    9    8    5
```

3) Write a simple Java thread program to compute the sum of n natural numbers. The program should read the number of threads m and value of n from the user. Each of the threads should add its share of assigned number to a global variable. When all the threads are done, the global variable should contain the result. The program should use a Synchronized block to make sure that only one thread is updating the global variable at a given time

```java
import java.util.Scanner;

class Global {
    public static int sum;

    public static synchronized void add(int n) {
        sum += n;
    }
}

class ChildThread extends Thread {
    int m, n;

    ChildThread(int n1, int n2) {
        m = n1;
        n = n2;
        start();
    }

    public void run() {
        for (int i = m; i <= n; i++)
            Global.add(i);
    }
}

class SumDemo {
    public static void main(String args[]) {
        int m, n, i;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the value of n:");
        n = sc.nextInt();
        System.out.println("Enter the number of threads:");
        m = sc.nextInt();
        ChildThread ct[] = new ChildThread[m];
        for (i = 0; i < m - 1; i++)
            ct[i] = new ChildThread((n / m) * i + 1, (n / m) * (i + 1));
        ct[i] = new ChildThread((n / m) * i + 1, n);
        try {
            for (i = 0; i < m; i++)
                ct[i].join();
        } catch (InterruptedException e) {
            System.out.print(e);
        }
        System.out.println("Sum of the " + n + "numbers:" + Global.sum);
    }
}
```

```
OUTPUT:
Enter the value of n:
5
Enter the number of threads:
3
Sum of the 5numbers:15
```

4) Write a Java thread program to search the minimum number in a given array. The program should read the number of elements in the array, number of threads to be created and the array elements from the user. Each thread should find minimum element in an assigned block of elements and compare to global minimum element. When all the threads are done, the global variable should contain the minimum element. It should use a Synchronized block to make sure that only one thread is updating the global minimum variable at any given time

```
import java.util.Scanner;

class Global {
    public static int min, tno;

    public static synchronized void update(int n) {
        if (tno == 0)
            min = n;
        else {
            if (n < min)
                min = n;
        }
        tno++;
    }
}

class ChildThread extends Thread {
    int m, n, arr[];

    ChildThread(int n1, int n2, int a[]) {
        m = n1;
        n = n2;
        arr = new int[n2 - n1 + 1];
        for (int i = 0; i < n - m + 1; i++)
            arr[i] = a[m + i];
        start();
    }

    public void run() {
        int small = arr[0];
        for (int i = 1; i < n - m + 1; i++) {
            if (arr[i] < small)
                small = arr[i];
        }
        System.out.println(m + "\t" + n + "\t" + small);
        Global.update(small);
    }
}

class MinDemo {
    public static void main(String args[]) {
        int m, n, i, arr[];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements of the array:");
        n = sc.nextInt();
        arr = new int[n];
        System.out.println("Enter the number of threads to be created:");
        m = sc.nextInt();
        ChildThread ct[] = new ChildThread[m];
        System.out.println("Enter " + n + " elements:");
        for (i = 0; i < n; i++)
            arr[i] = sc.nextInt();
        for (i = 0; i < m - 1; i++)
            ct[i] = new ChildThread((n / m) * i, (n / m) * (i + 1) - 1, arr);
        ct[i] = new ChildThread((n / m) * i, n - 1, arr);
        try {
            for (i = 0; i < m; i++)
                ct[i].join();
        } catch (InterruptedException e) {
            System.out.print(e);
        }
        System.out.println("Minimum element of the array : " + Global.min);
    }
}
```

OUTPUT:

```
Enter the number of elements of the array:
5
Enter the number of threads to be created:
3
Enter 5 elements:
1 2 3 4 5
2       4       3
1       1       2
0       0       1
Minimum element of the array : 1
```

NAME – PRIYANSHU MALLICK
SIC – 21BCSF11
ROLL. NO – 30
SEC – B
GROUP – B2
BRANCH - CSE