# **LAB SESSION 08**:

**MULTITHREADING APPLICATIONS**

**Date of the Session: / / Time of the Session: \_\_\_\_\_\_to**

**Pre-Lab:**

1. You are given a sequence A1,A2,…,AN*A*1,*A*2,…,*AN*. You need to determine if it is possible to choose two indices i and j such that Ai \neq Aj*Ai≠Aj*, but A[Ai] = A[Aj]*A*[*Ai*]=*A*[*Aj*]. (If it was possible, print RUN THREAD or else print STOP THREAD.)

Input:

-The first line of the input contains a single integer T*T* denoting the number of test cases. The description of T*T* test cases follows. -The first line of each test case contains a single integer N*N*. -The second line contains N space-separated integers A1,A2,…,AN*A*1,*A*2,…,*AN*.

Output:

For each test case, print a single line containing the string "RUN THREAD" if it is possible to choose required indices or "STOP THREAD" otherwise..

**Sample Input:**

4

4

1 1 2 3

4

2 1 3 3

5

5 4 4 3 1

5

3 2 1 1 4

**Sample Output:**

RUN THREAD

STOP THREAD

STOP THREAD

RUN THREAD

**Solution:**

**In-Lab:**

1. You have the five functions:

* printTwo that prints the word " divisible by 2" to the console,
* printThree that prints the word " divisible by 3" to the console,
* printFour that prints the word " divisible by 4" to the console,
* printFive that prints the word “divisible by 5” and
* printNumber that prints a given integer in other conditions.

 Implement all these functions by calling multiple threads to print the numbers from 1 to 15.

**Solution:**

1. Write a Java program using Synchronized Threads, which demonstrates Producer Consumer concept.

**Solution:**

1. You have a function printNumber that can be called with an integer parameter and prints it to the console.

For example, calling printNumber(7) prints 7 to the console.

You are given an instance of the class ZeroEvenOdd that has three functions: zero, even, and odd. The same instance of ZeroEvenOdd will be passed to three different threads:

Thread A: calls zero() that should only output 0's.

Thread B: calls even() that should only output even numbers.

Thread C: calls odd() that should only output odd numbers.

Modify the given class to output the series "010203040506..." where the length of the series must be 2n.

Implement the ZeroEvenOdd class:

ZeroEvenOdd(int n) Initializes the object with the number n that represents the numbers that should be printed.

void zero(printNumber) Calls printNumber to output one zero.

void even(printNumber) Calls printNumber to output one even number.

void odd(printNumber) Calls printNumber to output one odd number.

Example 1:

Input: n = 2  
Output: "0102"  
Explanation: There are three threads being fired asynchronously.  
One of them calls zero(), the other calls even(), and the last one calls odd().  
"0102" is the correct output.

Example 2:

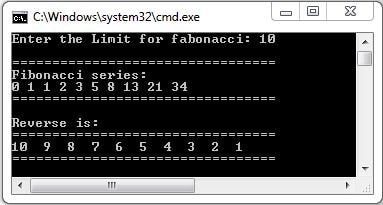
Input: n = 5  
Output: "0102030405"

**Solution:**

**Post-Lab:**

1. Write a JAVA program which will generate the threads:  
     
   - To display 10 terms of Fibonacci series.  
   - To display 1 to 10 in reverse order.

Sample output:



**Solution:**

(For Evaluator’s use only)

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Comment of the Evaluator (if Any) | Evaluator’s Observation  Marks Secured: \_\_\_\_\_\_\_ out of \_\_\_\_\_\_\_\_ Full Name of the Evaluator:  Signature of the Evaluator Date of Evaluation: | |

**Reference links:**

**Pre Lab:**

1. https://www.codechef.com/problems/BYTCD1

**In lab:**

1. https://leetcode.com/problems/fizz-buzz-multithreaded/
2. https://www.tutorialride.com/java-multithreading-programs/producer-consumer-problem-java-program.htm

**Postlab:**

**1.** https://www.tutorialride.com/java-multithreading-programs/print-fibonacci-reverse-series-with-thread-class.htm