BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE PILANI, K. K. BIRLA GOA CAMPUS

I SEMESTER 2015-2016

Advanced Operating Systems (CS G623)

Assignment 3

Due date 13/09/2015 (11:59 P.M)

Instructions:

(1) Please upload the assignment using **http://photon**
(The file name should be <your id number>.tar.gz Example: **2015H103015G.tar.gz**)

(2) This is an **individual** assignment. Please see section 4b of handout for Malpractice regulations.

(3) The programming assignments will be graded according to the following criteria

- Completeness; does your program implement the whole assignment?
- Correctness; does your program provide the right output?
- Efficiency; have you chosen appropriate algorithms and data structures for the problem?
- Programming style (including documentation and program organization); is the program well designed and easy to understand?
- Viva conducted by me.

DO NOT FORGET to include a README file (text only) in your tar.gz file with the following contents.

General README instructions

In the directory you turn in (please upload the assignment as a tar.gz file), you must have a text-only file called README, in which you will cover AT LEAST the following:

1. Your name. If you interacted significantly with others indicate this as well.
2. A list of all files in the directory and a short description of each.
3. HOW TO COMPILE your program.
4. HOW TO USE (execute) your program.
5. A description of the structure of your program.
6. In case you have not completed the assignment, you should mention in significant detail:
   - What you have and have not done
   - Why you did not manage to complete your assignment (greatest difficulties)
     This will allow us to give you partial credit for the things you have completed.
7. Document any bugs of your program that you know of. Run-time errors will cost you fewer points if you document them and you show that you know their cause. Also describe what you would have done to correct them, if you had more time to work on your project.

NB: I will remove the link exactly by 12 Mid night. Those who are taking the lifeline can mail the code (only one lifeline is allowed for the entire set of assignments) to biju@goa.bits-pilani.ac.in within 24 hours of the deadline.

Please refer section 4b of the handout to know more about Malpractice regulations of the course.

Question 1: Prime Number Generation

Implement an optimized prime number generation program with the help of threads. The main program must create a new thread (say prime), which will get value ‘n’ (Maximum limit of the prime numbers) from the user. The prime thread will create ‘n/2’ new threads (these threads should start finding whether the number is prime or not only after the prime thread finishes creation of all ‘n/2’ threads. The newly created threads check whether the number given to it is a prime number or not. Make sure you are creating all the threads which are multiple of 3, 5 and 7 as detachable and other threads as joinable. The threads should return status as 0 if it is not a prime and 1 if it is a prime. Printing of prime numbers must be done in the main thread (descending order). Each thread must print its Thread **ID** (tid) and process **ID** (pid). Make sure that your main thread is exiting only after the entire joinable threads exit. You are not allowed to use any global variables in this program.

Sample Result

My Thread ID is **2058** and my process ID is 4**050**

Please Enter Data: 10

My Thread ID is **2063** and my process ID is 4**050**

My Thread ID is **2062** and my process ID is 4**050**

Prime number: 7

My Thread ID is **2061** and my process ID is 4**050**

Prime number: 5

My Thread ID is **2060** and my process ID is 4**050**

Prime number: 3

My Thread ID is **2059** and my process ID is 4**050**

Prime number: 2

From Main thread

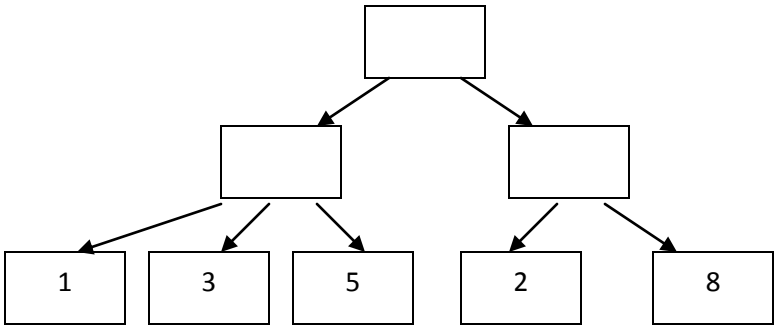
7 5 3 2

Program Exiting.

Question 2: Fibonacci Number Generation

Implement an optimized Fibonacci number generation program with the help of thread. The main program must create a new thread, which will get value 'n' (Maximum limit of the Fibonacci series, n is a local variable) from the user. This new thread will create 2 new threads as quickly as possible (use a global variable for it). One of the two newly created threads will create a new thread whenever it encounters a new odd Fibonacci number and the newly created thread will print the Fibonacci value. The other thread will create a new thread whenever it encounters a new even Fibonacci value and the newly created thread will print that value. The threads must communication between each other to print the Fibonacci numbers in ascending order. The main must print the resultant Fibonacci series. Make sure that the main thread is exiting only after all the threads exit. You are allowed to use any number of global variables in this program.

Tree Structure of newly created processes



Sample Result

My Thread ID is **2051** and my process ID is 4**050**

Please Enter Data: 10

My Thread ID is **2052** and my process ID is 4**050**

I am Odd Process

My Thread ID is **2053** and my process ID is 4**050**

I am Even Process

My Thread ID is **2054** and my process ID is 4**050**

Fib number: 1

My Thread ID is **2055** and my process ID is 4**050**

Fib number: 2

My Thread ID is **2056** and my process ID is 4**050**

Fib number: 3

My Thread ID is **2057** and my process ID is 4**050**

Fib number: 5

My Thread ID is **2058** and my process ID is 4**050**

Fib number: 8

1 2 3 5 8

Program Exiting.