Assignment for 3rd Year 1stSemester Students
IT/PC/B/S/313

GUIDELINES

1. Try to write a clean program with enough comments.
2. At the beginning of the file, use block comments to write details about name, roll no, assignment details, input required and output generated.
3. Also put the compilation [should be WARNING free] and execution sequence under the block comment.
4. The name of the file should be as per the following format.

   <Two Digit Team Number>_<Assignment Number>.c

5. The type of the file should be pure plain ASCII Text.
6. The assignment files should be uploaded AS PER THE LAB SCHEDULE. Upload only required no of files. NOT A BIT MORE.
7. While coding, always use indentation of 4 spaces.
8. Blocks of code should be separated by a newline.
9. Always use command line argument handling to take inputs.
10. Duplicate assignments will incur penalties. [Marks will be allocated proportionally]
11. Not adhering to any of these guidelines will incur penalties.
12. For the description of any system/library call use man command.
13. Always use ‘perror’ routine to check the return status of the system/library call.

ASSIGNMENT – 5-A1
Total Marks - 10 [CO4]
Creating DEADLOCK using Thread Programming

The objective of this assignment is to create a deadlock. For this purpose define two global variables (Total_1 and Total_2) and initialize both of them to 1000. You should also have two mutexes to protect these two global variables. You need to create two threads also.

The function of each of the threads is to generate a random quantity (not more than 50) and subtract that quantity from one of the Total and add that quantity to the other Total. While manipulating the Totals, each Thread should lock both the mutex and then unlock it after changing the Totals.

The order of locking and unlocking the Mutex is very important, because that's what creates a Deadlock. Once you correctly identify this order, you should upload the program. Also, include that information (as comment in your source file) how to avoid this kind of Deadlock.

If these two Threads, Two Totals and Two Mutex are not good enough for Deadlock; Then you need to create one more Thread, one more Total and and one more Mutex; and carry on the same experiment.

Make sure there are enough printf in your program so that it can be clearly understood that there is a Deadlock. One way to ensure enough printf is to display the total of Total_1 and Total_2 all the time after every operation on it.