CSE 301: Operating Systems
Homework 3
(due Noon Nov 5)

Instructions

1. The deadline is a hard one. The form upload will close at sharp noon Nov 5th. There will be no extensions.
2. Total marks = 6 , Potentially max. marks = 8
3. You have to type the assignment using a word processing engine, create a pdf and upload on the form. Please note that only pdf files will be accepted.
4. Name the submission as {branch}_{roll_number}_{name}.pdf
5. All code/Jupyter notebooks must be put up as **secret gists** and linked in the created pdf. Again, only secret gists. Not the public ones.
6. Any instances of cheating/plagiarism will not be tolerated at all.
7. Cite all the pertinent references in IEEE format.
8. The least count of grading would be 0.5 marks.
9. Some suggestions for plotting - WolframAlpha, Academo.Org, Geogebra, Matploltib, GNUplot, Matlab, Octave
10. You can find the course VM on the course web page. The root password is: 1234

1. Implement a concurrent B-tree with a simple locking strategy such as a single lock. There are three main operations - search, insert and delete. Measure its performance as the number of concurrent threads increases. Make a simple line plot with x axis as number of threads and y axis as time taken for N operations. Repeat the experiment multiple times to ensure robustness of results.

   What is the CPU configuration of your laptop? Can you explain the plot

in terms of the number of CPU cores and threads.[**3 marks**]

2. Bonus: For above question, let's assume we know that search is called a lot more than insert and delete. Read about readers, writer lock and add the functionality that multiple readers can read concurrently. [**2 marks**]

3. Write a C program to implement dining philosopher's problem using condition variables and mutex, without using semaphores. [**1.5 marks**]

4. Compare the following solution to dining philosopher's problem with the solution given in the text. The comparison has to be done in terms of deadlock avoidance, concurrency and avoiding starvation. You can consider various scenarios, like, 1) eating takes much more time than picking the forks, and any philosopher can think for any length of time; 2) eating is comparable to picking up forks, etc..[**1.5 mark**]

```
1 void getforks() {
2  mutex_lock(&m)
3  sem_wait(forks[left(p)]);
4  sem_wait(forks[right(p)]);
5  mutex_unlock(&m)
6 }
```