REPORT

I. Some important points

- default_random_engine is used to generate random numbers from exponential_distribution to simulate critical and remainder sections . Parameters for both critical and remainder section were taken 1/λ1 and 1/λ2 respectively.
- We have used usleep function to make threads sleep for time given by exponential distribution in microseconds. It is available in unistd.h header file.
- To get time when thread requested, entered and exited critical section, we have used time_t functionality available in time.h header file.
- We have used inbuilt exponential distribution functionality in the code. It is available in random header file.
- string getTimeinReqdFormat(time_t) to change the real time in the format hr:min:sec.
- workingLock is the lock variable used in all the three programs.
- We have used inbuilt TAS, CAS functions present in atomic header file.
  TAS – test_and_set()
  CAS – compare_exchange_strong()

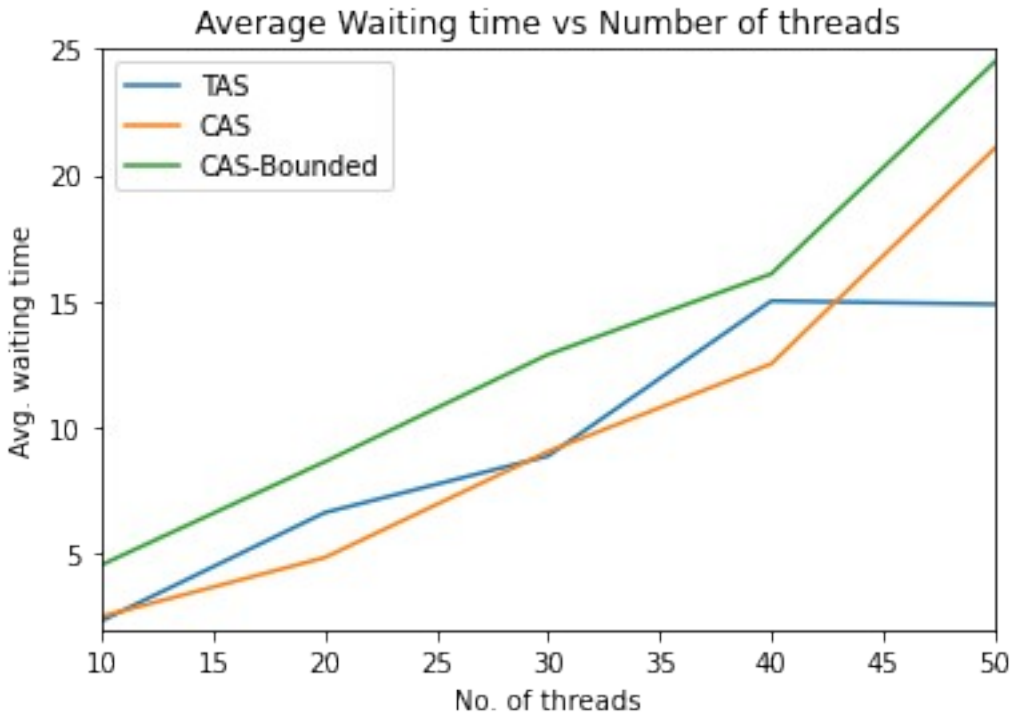
- CAS Bounded makes use of a shared array waiting that keeps track of which thread is waiting. It ensures bounded waiting time for each thread. The waiting array is made atomic to deal with concurrency issues that may arise.

II. Graphs

- Parameters taken are:

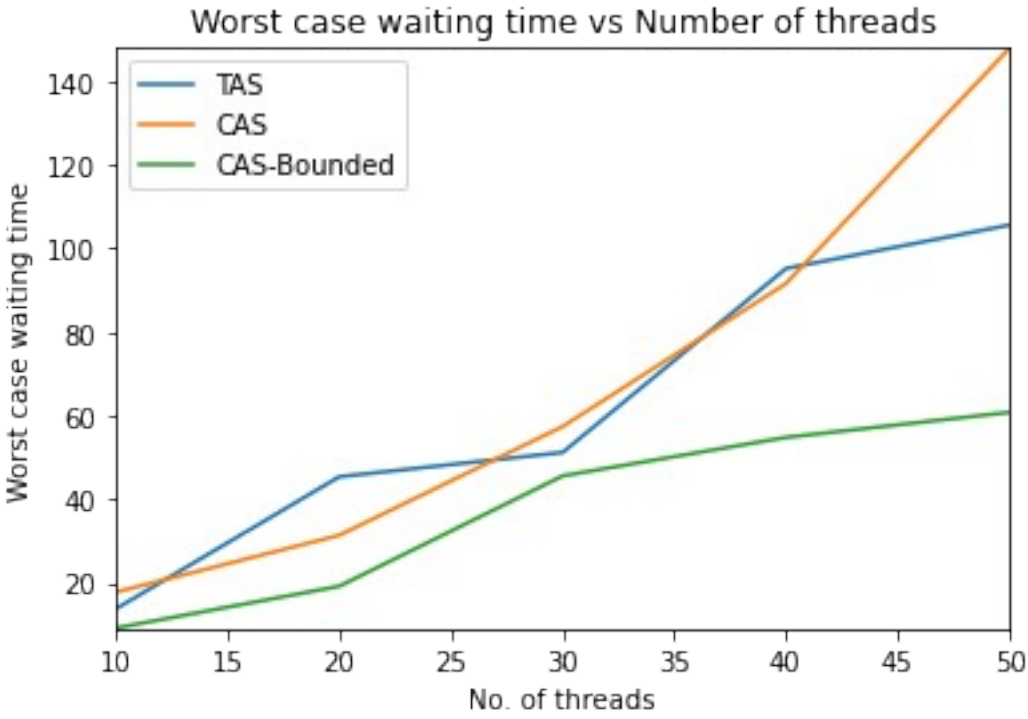
  n varying from 10 to 50
  k = 10
  lambda1 = 0.4
  lambda2 = 0.6

- Average Waiting time vs Number of threads



The average waiting time of TAS and CAS are roughly same . Average waiting time taken by Bounded-CAS is greater than both TAS and CAS as it ensures that no thread starves. We observe that Average waiting time roughly increases with increase in number of threads in case of all the three algorithms.

- Worst case waiting time vs Number of threads



Worst waiting time taken by the TAS and CAS algorithm are very very close to each other. CAS bounded performs better in terms of worst waiting time as it gives thread almost equal chance to enter CS and thus avoids starvation. Because of the above reason, CAS has less worst case waiting time than TAS and CAS.