OPERATING SYSTEMS 2
PROGRAMMING ASSIGNMENT 2

I- Design of the program

Critical Section :
A segment of code which we want only a single thread to access at a given time is called a critical section.

Entry Section :
A part of code which is available to all the threads where the threads contend to enter into the Critical Section. Bounded wait guarantees that no thread waits forever in the entry section and gets a fair chance to enter Critical section.

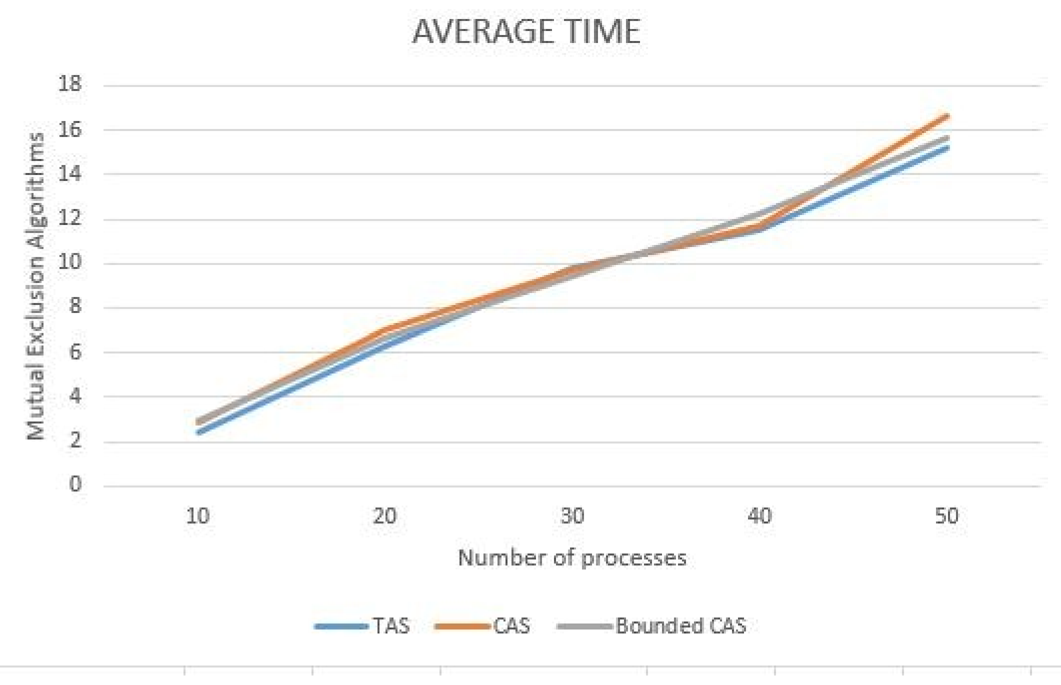
Exit Section :
A segment of code wherein a thread exits from the critical section and opens the lock for other threads to enter the CS.

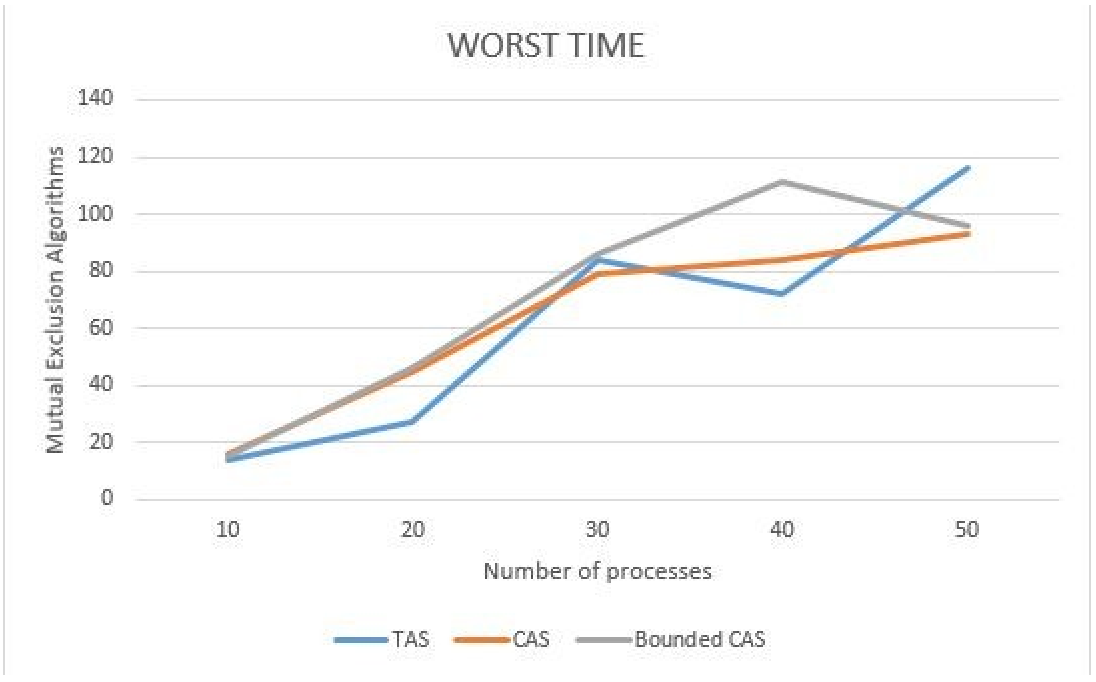
Important Libraries Used :
- <pthread.h> : Used for creation and work assignment of the threads.
- <atomic> : Used for making the lock atomic so that only one thread can edit it at a time.
- <unistd.h> : Used for calculating the current system time.

II-IMPLEMENTATION OF THE PROGRAM

All the three programs follow a similar design , except the implementation of the critical section. The main thread reads the input from “inp-params.txt”. The main thread then creates n threads, each of which executes testCS function. Each thread, in testCS function, simulates Entry section, Critical Section and Remainder section. Each thread also measures the time at which it entered all of these sections and also measures the waiting time (time spent in entry section before entering the critical section). The lock variable(std::atomic_flag locki = ATOMIC_FLAG_INIT) is an atomic variable used by the standard library functions for mutual exclusion. TAS and CAS use the standard library functions atomic_flag_test_and_set, and atomic_compare_and_exchange to implement critical section. CAS-bounded uses additional local variable j, and a global array waiting to allow for fairness - i.e. each thread should get a chance to enter the critical section. It reduces the randomness in selecting the next thread to send in the critical section by selecting the thread which has thread_id linearly greater than the thread in critical section , namely j .

GRAPHS:





For Exponential Distribution
Lambda1 = 0.8
Lambda 2= 0.5
n=(10-50)
k=10