OS Assignment 3
Deadline : 29th October 2359Hrs

1. Producer Consumer Problem :

Given the size of a circular buffer and number of readers, write a code with n+1 threads, n for readers and 1 for the writers. All are accessing the same circular buffer. The following conditions should be satisfied:

- A reader cannot read in an empty location.
- A reader will not read from the same location twice if the writer has not overwritten.
- A writer cannot overwrite a place which has not been read by every reader.

Input : Size of buffer, n
You can use random function for writing data into the buffer.

2. The War of Roses :

The conflict between the Yorks and Lanncasters in Britain is famously remembered as the War of Roses. Though both are descendants from the same bloodlines, they were invloved in a massive struggle for power which lead to many wars.

Inn keepers have a real tough time during the wars, especially those whose inn lies in the battle zone. But they are also the only source of good ale, and thus both the armies don't want to burn em down. Hence, both the parties came to a treaty. The treaty was that at a given time no two soldiers of different armies can be present inside the inn at the same time. Also, since the size of the inn is not infinite, after the inn is full, others shall wait outside in line, without making any conflict. They are encouraged to share songs amongst themselves, and have a laugh while they wait.

Each soldier is a thread, and is randomly to be assigned as a York or a Lanncaster.
The capacity of the inn will be given as input.
Simluate the above given scenario, without causing any conflicts to the treaty.
Make sure that no soldier has to wait unnecesarily outside.

3. Concurrent Merge Sort in Shared Memory

Given a number n and n numbers, sort the numbers using Merge Sort.

Recurssively make two child processes, one for the left half, one of the right half. If the number of elements in the array for a process is less than 5, perform a selection sort. The parent of the two children then merges the result and returns back to the parent and so on.

Compare the performace of the merge sort with a normal merge sort implementation and make a report.

“You must use the shmget, shmat functions as taught in the tutorial.”

4. Perf Testing

The following 2 questions are open ended. Marks in this question will be given relative to the most exhaustive report and testing.

Make use of $ perf stat -p <pid> sleep <seconds>

a) Dtella perf testing

Run dtella on your machines and connect to the hub on DC client of your choice for 3-4 hours. Run the perf test on the dtella every 10/20 seconds and make a plot of :

1. CPU-usage vs Time
2. Context Switch per second vs Time

An exhaustive testing will imply running the dtella with various other programs which are I/O intensive or Cpu intensive or both and see how the graphs vary, etc.

Follow this link to download and install dtella: https://github.com/ffledgling/dtella

You can use any plotting library to plot the graph.

b) Merge Sort of Big File (>1GB) using bash command sort.

Generate a file > 1GB in size containing integers and run the sort command on bash to sort it while running perf on it.

Write a script which splits the file into two parts recursively and then runs the sort command once the size of the split is at below some threshold, say 10MB, 100MB or 200MB and so on, and then performs a merge operation.

You must show results for atleast 3 different and dissimilar thresholds.

Run perf on the script you write for each of the thresholds chosen and compare how the results of perf vary with increase in the threshold size, or number os splits.

You are free to write the script in any language of your choice.