**Question#1**

Write a C program that uses semaphores to implement a producer-consumer problem with two threads. The program should create two threads, one for the producer and one for the consumer, and use semaphores to synchronise access to a shared buffer. The producer should write random integers to the buffer and the consumer should read them and print them to the console.

**Question#2**

Write a C program that uses the fork system call to create a child process that executes a shell command. The parent process should wait for the child process to finish and then print the exit status to the console. The shell command should be specified by the user at runtime.

**Question#3**

You are given a matrix of integers. Use the **pthreads** library to assign threads to calculate the sum of each row (1 thread per row). Find the row with the maximum sum of elements. If there are multiple rows with the same maximum sum, output the row with the smallest index.

**Question#4**

You are given a list of integers. Write a program that creates two threads. The first thread should calculate the sum of all the even numbers in the list, while the second thread should calculate the sum of all the odd numbers in the list. Use semaphores to ensure that the second thread doesn't start until the first thread has finished its calculations. Once both threads have finished their calculations, print out the sum of the even numbers and the sum of the odd numbers.

**Question#5**

You are given a list of 100000 integers and an integer k. Your task is to find the kth largest element in the list. Optimise your solution using OpenMP libraries to divide the workload into threads.

**Question#6**

Write a program that simulates a game of rock-paper-scissors between two players. The program should ask the user for the number of games to be played and for the choice of each player in each game. Then, the program will run the game by creating 2 child processes and having them communicate their option (rock,paper,scissor) to the parent process. The parent process will then judge which player is the winner of the game. This will be repeated N number of times, where N is the number of games to be played (user input). The program should output the winner of each game and the overall winner of the series.

**Question#7**

Write a program that simulates a race between two runners. The runners move along a track, and at each time step, they randomly move forward a certain distance. The program should create two threads, one for each runner, and use semaphores to ensure that they take turns moving. The first runner to reach the end of the track wins the race.

**Question#8**

You are given a string of words, separated by spaces. Write a program that reverses the **order** of the words in the string. The program should ask the user for the string, divide the string into equal sized blocks and assign these to different threads using **pthreads** library. Each thread will reverse the order of the block of words assigned to it. Finally, the main process will compile all the data together and print the final string which will be in reversed order.

**Question#9**

You are given a list of strings, each string is considered a sentence. Write a program that takes as input the list of sentences and a target word, and outputs a list of all the sentences that contain the target word. The program should be case-insensitive, i.e., it should match the target word regardless of whether it appears in uppercase or lowercase. Divide the workload by using OpenMP library.

**Question#10**

You are given a list of strings. Write a program that creates a thread for each string in the list. Each thread should count the number of vowels in the corresponding string and update a shared counter variable. Use semaphores to ensure that the counter variable is updated correctly. Once all threads have finished their calculations, print out the total number of vowels in the entire list.

**Question#11**

Write a program to implement a simple chat room application. The main process will act as the server, while there will be N threads which will behave as the users. When a user wants to send a message in the chat room, it will communicate with the server and the server will update the chat room (shown on the console screen).

**Question#12**

Write a program to simulate a game of Tic Tac Toe between two players. The program should print the current board after each move, and should check after each move whether a player has won or the game has ended in a draw. The program should continue until a player has won or the game has ended in a draw. The program should be generalized to work for any size of the board (NxN), and for any number of players. Create 1 thread for each player, and use semaphores to apply locks to the Tic Tac Toe board, to ensure that only 1 player will move at 1 time.

**Question#13**

Write a C program that uses the **OpenMP** library to implement a multithreaded matrix multiplication algorithm. Make the program generalised for square matrices of any size. The program should prompt the user for the size of the matrices and then allocate memory for them dynamically. It should then create multiple threads to perform the matrix multiplication in parallel and print the result to the console.