**Calculating the average of a list of integers by exploiting parallelism:**

Suppose we have an array of integers: 1, 10, 9, 100, 23, 4, 11, 12, 3, and 1. The size of this array is 10. Now if we want to calculate the average, we can simply do:

Average = sum of elements/ total elements.

Now, if we have a large number of elements, then it will take a lot of time to calculate the average. We can utilize parallelism by dividing the task of finding the sum among different processes. Suppose a process p1 finds the sum of first 4 elements:

Sum1=1+10+9+100= 120

Count=4

And another process p2 calculates the sums of remaining 6 elements:

Sum2=23+4+11+12+3=53

Count=6

Then average calculated as:

Total Sum = Sum1+Sum2= 120+53= 173

Total Count= Count1 + Count2= 4+6=10

Average= Total Sum/Total Count= 173/10= 17.3

**Question 1.** Suppose there are two processes. Each process reads a different file having a list of integers. Both processes read the integers, calculate their sum, and send the sum and the count of integers to a server process via shared memory. The server then finds the total average by following formulae:

Total Sum= sum of integers sent by p1 + sum of integers sent by p2

Total Count= count sent by p1 + count sent by p2

Average= Total Sum/ Total Count

After calculating the average, the server sends the average to both processes. Both processes then print the sum on their respective terminal. (You need to synchronize the processes using semaphores on shared memory)

**Shared Memory Portions Required:**

1. The will be a single shared memory portion on which both processes will place their (sum, count) pair. One process will put the pair on 0th index and the other will put the pair on 1st index.
2. A shared memory portion for current available index number where a process can put the pair.
3. A shared memory portion for placing the average by server. (Only one portion required which will be shared by both worker processes)
4. Shared memory portions for the required semaphores.

**Question 2:**

Write C++/C code for a program that takes as command line argument a source file name. The program creates a producer thread which reads each time 20 character from the file and writes the characters to a shared memory buffer. The shared memory buffer can store at most 20 characters. Now there is another thread which acts as a consumer, and reads those 20 characters from shared memory, prints the characters on the screen, and waits for the user to press enter key. The producer can only write the next 20 characters of file to shared memory after the consumer has read the previous 20 characters from the shared memory. So, you need to synchronize the two threads using semaphores. When the file has been complete written to the shared memory, the producer writes a $ to shared memory which indicates to the consumer that file has been finished. (Assume the file’s size is always multiple of 20, such as 20, 40, 60, etc.)

**For example:** Suppose there is a file of 40 characters.

* Producer writes 20 characters and waits for the consumer to read these 20 characters.
* Consumer reads 20 characters, prints on the screen and waits for user to press enter key.
* Producer now writes the next 20 characters and waits for the consumer to reads the 20 characters.
* The consumer reads the 20 characters, prints the data on the screen, and waits for the user to press enter key.
* The end-of-file has been reached, so the producer writes $ to the shared memory.
* The consumer reads $, which indicates that the file’s data has completely been read.