Q1) Hello World Program

- The default number of threads created is equal to the number of cores in the system.

Q2) Hello World Program - Version 2

- The printHello() is called with a threadID.

Q3) DAXPY

- The DAXPY operation is experimented with 1-32 threads.
- The speedup is calculated for each of the experiments.
- The max speedup is reported for different number of threads.
- Maximum speedup is obtained for 5 threads.
- Increasing the threads beyond 5 decreases performance as the thread performance may be limited owing to CPU resources, memory resources. Increase in overhead may also be due to context switching.

Q4) Matrix Multiplication

- The worksharing for loop construct partitions the iterations required for matrix multiplication operations

Q5) Calculation of PI

- Shared variables - num_steps, step, sum
- Thread private variables - interval size, low, high, thread_ID, partial_sum
- The program spawns 4 threads (although an option to input the number of required threads is available).
- The calculation of PI has been carried out using two approaches:
    1. **Interval Approach -** The interval 0-1 is divided into 100000 sub-intervals. Low and upper limits for the iterations carried out by each of the threads are calculated and partial sums are obtained. The total sum is updated atomically.
    2. **Mod Approach -** The iterations are allocated to each of the thread in a round-robin fashion i.e a thread with thread_ID = ID is assigned all iterations where iteration_number % (number of threads) = ID. The total sum is calculated from the partial sums in the critical region which provides the required Mutual Exclusion.

Q6) Calculation of PI - Worksharing and Reduction

- Worksharing **for** construct splits up the loop iterations among the threads in a team. (Note: The worksharing **for** construct could be used in this problem as the loop iterations do not have any dependencies).
- Reduction can be used for shared variables that may be subjected to race condition. It ensures that the shared variables are updated atomically.

Q7) Calculation of PI - Monte Carlo Simulation

- The LCG (linear congruential generator) is used for pseudo random generation.
- Seed values are calculated for each of the thread numbers to ensure
- LEAP FROG method is used to ensure that the same sequence of values are obtained regardless of the number of threads

- The value of PI is estimated to be equal to the ratio of the circle and dart board areas, and is obtained as 3.1453

Q8) Producer Consumer Program

- **Flush** guarantees a consistent view of memory with respect to the list of variables mentioned in the flush set. The action of Flush ensures that
    1. All R,W operations that overlap the flush set and occur prior to the flush complete before the flush executes.
    2. All R,W operations that overlap the flush set and occur after the flush don't execute until after the flush.
    3. Flushes with overlapping flush sets cannot be reordered.
- The main logic behind the producer consumer problem is that a consumer should consume only after the producer finishes producing.
- This requires pairwise synchronization as we have two threads namely the producer thread and consumer thread that need to coordinate in order to solve the problem.
- In order to achieve the desired pairwise synchronization we use a shared **flag** variable. The reader (consumer) spins waiting for the value of the flag (i.e stays in an infinite loop till the value of flag gets updated).
- Appropriate flushes are used to force updates to and from the memory.