**Question 1:** Provide three programming examples in which multithreading provides better performance than a single-threaded solution.

* **Web crawling - multiple threads may be utilized to handle concurrent http requests at one time, increasing the rate at which web data can be scraped.**
* **Game design - A main thread may be used to handle the core game loop, and a separate thread may be used for some subroutine (such as a physics simulator or NPC algorithms) in order to keep the core loop from interrupting.**
* **Machine learning - multiple threads can be utilized to train a model on multiple parts of a dataset simultaneously, effectively dividing training time by the number of threads used for training.**

**Question 2:** Using Amdahl's Law, calculate the speedup gain of an application that has a 60 percent parallel component for (a) two processing cores and (b) four processing cores.

1. S = 1 / (1-**0.6**+(**0.6**/**2**)) = 1.42 = **42% speed increase**
2. S = 1 / (1-**0.6**+(**0.6/4**)) = 1.82 = **82% speed increase**

**Question 3:** Write a Java program using Runnable interface to print even and odd numbers using two threads.

Included; also in **even\_odd.java** in submission

| public class Main {  public static void main(String args[]) {  Print printRunnable = new Print();  Thread t1 = new Thread(new PrintNumbers(false, printRunnable));  Thread t2 = new Thread(new PrintNumbers(true, printRunnable));    t1.start();  t2.start();  } }  class Print {  private Object lock = new Object();  private volatile boolean isEven = true;    public void even(int n) throws InterruptedException {  synchronized(lock) {  while (isEven) {  lock.wait();  }  System.out.println(n);  isEven = true;  lock.notifyAll();  }  }    public void odd(int n) throws InterruptedException {  synchronized(lock) {  while (!isEven) {  lock.wait();  }  System.out.println(n);  isEven = false;  lock.notifyAll();  }   } }  class PrintNumbers implements Runnable {  private boolean isEven;  Print print;    PrintNumbers(boolean isEven, Print p) {  this.isEven = isEven;  this.print = p;  }    public void run() {  int n;  if (isEven) {  n = 2;  } else {  n = 1;  }    while(n <= 20) {  if(!isEven) {  try{print.odd(n);} catch (InterruptedException e) {}  } else {  try{print.even(n);} catch (InterruptedException e) {}  }  n += 2;  }  } } |
| --- |