Task1

```
static class Counter {
    int count;
    void inc() {
        count = count+1;
    }
    int getCount() {
        return count;
    }
}
```

Write a thread class that will repeatedly call the `inc()` method in an object of type *Counter*. The object should be a shared global variable. Create several threads, start them all, and wait for all the threads to terminate. Print the final value of the counter, and see whether it is correct.
Let the user enter the number of threads and the number of times that each thread will increment the counter. You might need a fairly large number of increments to see an error. And of course there can never be any error if you use just one thread. Your program can use `join()` to wait for a thread to terminate

Task2
Write a program that will repeatedly read integer until 0. Create another class that implements Runnable interface. Find out the sum of all integers using a thread. **Note:** The sum algorithm should not be in a main class

Task3
Create an ArrayList of integers with the size 10000 and fill it with a random numbers. Calculate the number of all prime numbers and calculate the time of this algorithm. Create another two threads. First thread should work with first half of ArrayList and the second thread with the second part of ArrayList. Then find out the total number of prime number and compare the time of this algorithm with the previous one.

Task4
find the integer in the range 1 to 10000 that has the largest number of divisors. Now write a program that uses multiple threads to solve the same problem, but for the range 1 to 100000. By using threads, your program will take less time to do the computation when it is run on a multiprocessor computer. At the end of the program, output the elapsed time, the integer that has the largest number of divisors, and the number of divisors that it has.