1.Viết chương trình tạo 2 tuyến: một tuyến tìm kiếm các số nguyên tố từ 1000 đến 1000000 và một tuyến tính tổng giá trị của các số nguyên tố tìm được. Chú ý đồng bộ tuyến.

T1 tìm ra 1-2 , T2 sum 1-2,

T1 tim ra 123, T3 Sum 123

2. Viết chương trình tạo mảng có 1.000.000 phần tử, sau đó tạo 2 tuyến để sắp xếp 2 nửa mảng, cuối cùng ghép 2 mảng đã sắp xếp. So sánh cách làm trên với cách sắp xếp trực tiếp toàn bộ mảng.

Tính time chạy cho 2 phần -> kết luận

Code Tham khảo

Sắp xếp mảng

```
public class SapXepThread {
    private static final int ARRAY_SIZE = 400;
  private static final int NUM_THREADS = 4;

  public static void main(String[] args) {
    int[] array = createArray();
    System.out.println("Chuỗi Truóc SX: " + Arrays.toString(array));

    Thread[] threads = new Thread[NUM_THREADS];
    int segmentSize = ARRAY_SIZE / NUM_THREADS;

    for (int i = 0; i < NUM_THREADS; i++) {
      int startIndex = i * segmentSize;
      int endIndex = (i == NUM_THREADS - 1) ? ARRAY_SIZE - 1 : (startIndex + segmentSize - 1);
      threads[i] = new Thread(new SortTask(array, startIndex, endIndex));
      threads[i].start();
    }

    for (Thread thread: threads) {
      try {
        thread.join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }

    mergeSort(array, 0, ARRAY_SIZE - 1);

    System.out.println("Đã SX: " + Arrays.toString(array));
  }
```

```java
private static int[] createArray() {
  int[] array = new int[ARRAY_SIZE];
  for (int i = 0; i < ARRAY_SIZE; i++) {
    array[i] = (int)(Math.random() * 400); // Generate random numbers between 0 and 400
  }
  return array;
}

private static void mergeSort(int[] array, int left, int right) {
  if (left < right) {
    int mid = (left + right) / 2;
    mergeSort(array, left, mid);
    mergeSort(array, mid + 1, right);
    merge(array, left, mid, right);
  }
}

private static void merge(int[] array, int left, int mid, int right) {
  int[] temp = new int[right - left + 1];
  int i = left, j = mid + 1, k = 0;

  while (i <= mid && j <= right) {
    if (array[i] <= array[j]) {
      temp[k++] = array[i++];
    } else {
      temp[k++] = array[j++];
    }
  }
```

```
    while (i <= mid) {
      temp[k++] = array[i++];
    }

    while (j <= right) {
      temp[k++] = array[j++];
    }

    System.arraycopy(temp, 0, array, left, temp.length);
  }

  static class SortTask implements Runnable {
    private int[] array;
    private int startIndex;
    private int endIndex;

    public SortTask(int[] array, int startIndex, int endIndex) {
      this.array = array;
      this.startIndex = startIndex;
      this.endIndex = endIndex;
    }

    @Override
    public void run() {
      Arrays.sort(array, startIndex, endIndex + 1);
    }
  }

}
```

Số Nguyên Tố

```
private static final int NUM_THREADS = 5;
public static void main(String[] args) {
  int limit = 10000000;

  Thread[] threads = new Thread[NUM_THREADS];
  PrimeSumTask[] tasks = new PrimeSumTask[NUM_THREADS];

  int segmentSize = limit / NUM_THREADS;
  int start = 2;
  int end = segmentSize;

  long startTime = System.currentTimeMillis();

  for (int i = 0; i < NUM_THREADS; i++) {
    if (i == NUM_THREADS - 1) {
      // Last thread takes care of remaining numbers
      end = limit;
    }
    tasks[i] = new PrimeSumTask(start, end);
    threads[i] = new Thread(tasks[i]);
    threads[i].start();

    start = end + 1;
    end += segmentSize;
  }
  long sum = 0;
  for (int i = 0; i < NUM_THREADS; i++) {
    try {
      threads[i].join();
      sum += tasks[i].getSum();
    } catch (InterruptedException e) {
      e.printStackTrace();
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        long endTime = System.currentTimeMillis();
            System.out.println("Số Luồng = "+NUM_THREADS);
        System.out.println("Tổng các số nguyên tố đến " + limit + ": " + sum);
        System.out.println("Thời gian thực hiện: " + (endTime - startTime) + " milliseconds");
    }
    static class PrimeSumTask implements Runnable {
        private int start;
        private int end;
        private long sum;

        public PrimeSumTask(int start, int end) {
            this.start = start;
            this.end = end;
            this.sum = 0;
        }

        public long getSum() {
            return sum;
        }

        private boolean isPrime(int number) {
            if (number < 2) {
                return false;
            }

            for (int i = 2; i <= Math.sqrt(number); i++) {
                if (number % i == 0) {
                    return false;
                }
```

```
    private boolean isPrime(int number) {
      if (number < 2) {
        return false;
      }

      for (int i = 2; i <= Math.sqrt(number); i++) {
        if (number % i == 0) {
          return false;
        }
      }

      return true;
    }

    @Override
    public void run() {
      for (int i = start; i <= end; i++) {
        if (isPrime(i)) {
          sum += i;
        }
      }
    }
  }
}
```

3. demo đồng bộ 2 classs

```java
public class SynDongBoLuong {
    static Data dataShare;
    public static void main(String[] args) {
        dataShare=new Data();
        // Gán Cờ Phân chia công việc theo thứ tự
        //Quy ước 1 =L1 chạy,2=L2 và 3= Sum
        dataShare.setLaCo(1);
        // Luồng 1
        Thread T1=new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized(dataShare){
                for (int i = 0; i < 10; i++) {
                    try {
                        if (dataShare.getLaCo()==1) {
                        int a=new Random().nextInt(100);
                        dataShare.setA(a);
                        System.out.println("A= "+dataShare.getA());
                        Thread.sleep(90);
                        dataShare.setLaCo(2);//Gán Cờ cho luồng 2
                        dataShare.notifyAll();// đánh thức các luồng đnag ngủ dậy
                        }
                        else{
                            dataShare.wait();// Dừng- Đợi
                        }
                    } catch (InterruptedException ex) {
                        Logger.getLogger(SynDongBoLuong.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }}
        });
```

```java
public class SynDongBoLuong {
    static Data dataShare;
    public static void main(String[] args) {
        dataShare=new Data();
        // Gán Cờ Phân chia công việc theo thứ tự
        //Quy ước 1 =L1 chạy,2=L2 và 3= Sum
        dataShare.setLaCo(1);
        // Luồng 1
        Thread T1=new Thread(new Runnable() {
            @Override
            public void run() {
                synchronized(dataShare){
                for (int i = 0; i < 10; i++) {
                    try {
                        if (dataShare.getLaCo()==1) {
                        int a=new Random().nextInt(100);
                        dataShare.setA(a);
                        System.out.println("A= "+dataShare.getA());
                        Thread.sleep(90);
                        dataShare.setLaCo(2);//Gán Cờ cho luồng 2
                        dataShare.notifyAll();// đánh thức các luồng đnag ngủ dậy
                        }
                        else{
                            dataShare.wait();// Dừng- Đợi
                        }
                    } catch (InterruptedException ex) {
                        Logger.getLogger(SynDongBoLuong.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            }}
        });
```

Class data

```
12  public class Data {
13      private int a;
14      private int b;
15      private int LaCo;
16
17      public int getLaCo() {
18          return LaCo;
19      }
20      public void setLaCo(int LaCo) {
21          this.LaCo = LaCo;
22      }
23      public int getA() {
24          return a;
25      }
26      public void setA(int a) {
27          this.a = a;
28      }
29      public int getB() {
30          return b;
31      }
32      public void setB(int b) {
33          this.b = b;
34      }
35      public int TinhTong(){
36          return a+b;
37      }
38  }
39
```