There are the following JAVA classes implementing the multi-frontal direct solver algorithm for finite difference problem:

- *A.java*
- *A1.java*
- *A2.java*
- *AN.java*
- *Aroot.java*
- *BS.java*
- *E2.java*
- *Eroot.java*
- *Executor.java*
- *Main.java*
- *P1.java*
- *P2.java*
- *P3.java*
- *Production.java*
- *Vertex.java*

We have already implemented all the classes reponsible for matrix transformations. We already have the *Executor* class constructing the elimination tree:

```
  import java.util.concurrent.BrokenBarrierException;
  import java.util.concurrent.CyclicBarrier;
class Executor extends Thread {
  public synchronized void run() {
    barrier barrier = new barrier(this);
    Vertex S = new Vertex(null,null,null,''S'');
    try {
    //[(P1)]
    CyclicBarrier barrier = new CyclicBarrier(1);
    P1 p1 = new P1(S,barrier);
    p1.start();
    barrier.await();
    //$[(P2)_1(P2)_2]$
    barrier = new CyclicBarrier(2);
    P2 p2a = new P2(p1.m_vertex.m_left,barrier);
    P2 p2b = new P2(p1.m_vertex.m_right,barrier);
    p2a.start();
    p2b.start();
    barrier.await();
    //$[(P2)_3(P2)_4(P3)_5(P3)_6]$
    barrier = new CyclicBarrier(4);
    P2 p2c = new P2(p2a.m_vertex.m_left,barrier);
    P2 p2d = new P2(p2a.m_vertex.m_right,barrier);
    P3 p3a = new P3(p2b.m_vertex.m_left,barrier);
    P3 p3b = new P3(p2b.m_vertex.m_right,barrier);
    p2c.start();
    p2d.start();
    p3a.start();
    p3b.start();
    barrier.await();
    //$[(P3)_1(P3)_2(P3)_3(P3)_4]$
    barrier = new CyclicBarrier(2);
    P3 p3c = new P3(p2c.m_vertex.m_left,barrier);
    P3 p3d = new P3(p2c.m_vertex.m_right,barrier);
    P3 p3e = new P3(p2d.m_vertex.m_left,barrier);
    P3 p3f = new P3(p2d.m_vertex.m_right,barrier);
    p3c.start();
    p3d.start();
    p3e.start();
    p3f.start();
    } catch (InterruptedException | BrokenBarrierException e) {
      e.printStackTrace(); }
  }
}
```

We can execute now graph grammar productions reponsible for execution of the solver algorithm, see Figure 1

Please complete the *Executor* class by adding the graph grammar produc-

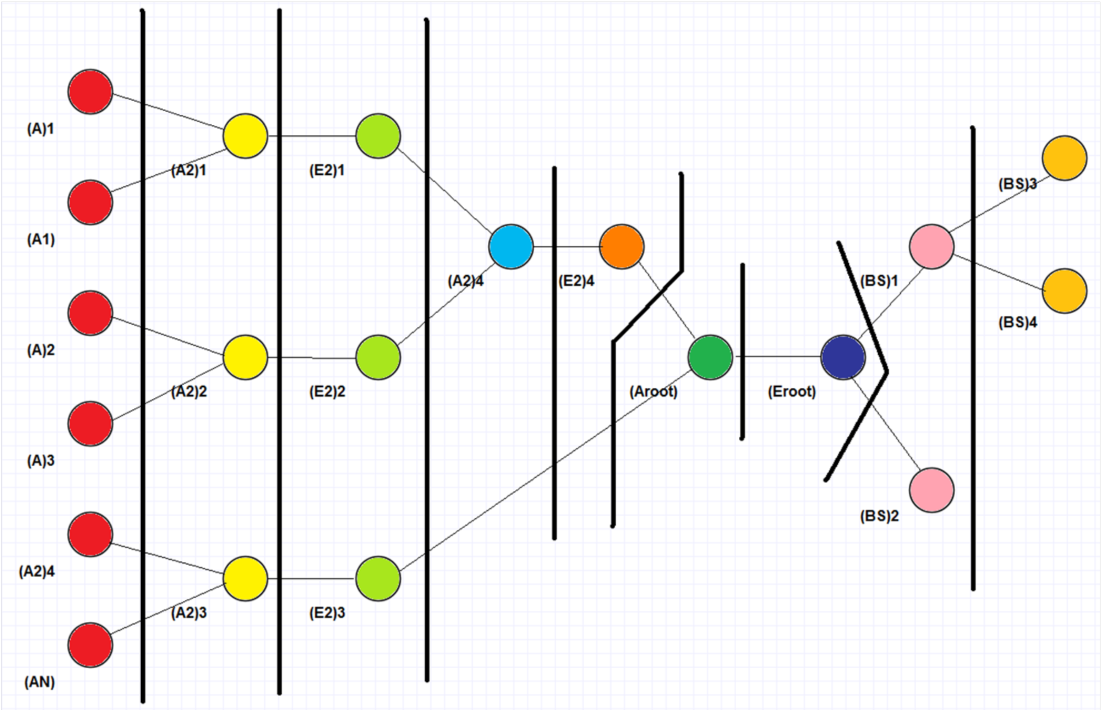

Figure 1: Graph of tasks responsible for execution of the solver algorithm.

tions responsible for solver execution.

Hint:

```
class Executor extends Thread {
  public synchronized void run() {
    // CONSTRUCTION OF ELIMINATION TREE
    ...
    // MULTI-FRONTAL SOLVER ALGORITHM
    //$[(A1)(A)_1(A)_2(A)_3(A)_4(AN)]$
    barrier = new CyclicBarrier(6);
    A1 localMat1 = new A1(p3c.m_vertex, barrier);
    A localMat2 = new A(p3d.m_vertex, barrier);
    A localMat3 = new A(p3e.m_vertex, barrier);
    A localMat4 = new A(p3f.m_vertex, barrier);
    A localMat5 = new A(p3a.m_vertex, barrier);
    AN localMat6 = new AN(p3b.m_vertex, barrier);
    localMat1.start(); localMat2.start(); localMat3.start();
    localMat4.start(); localMat5.start(); localMat6.start();
    barrier.await();
    //$[(A2)_1(A2)_2(A2)_3]$
    barrier = new CyclicBarrier(3);
    A2 mergedMat1 = new A2(p2c.m_vertex, barrier);
    A2 mergedMat2 = new A2(p2d.m_vertex, barrier);
    A2 mergedMat3 = new A2(p2b.m_vertex, barrier);
    mergedMat1.start(); mergedMat2.start(); mergedMat3.start();
    barrier.await();
    // PLEASE CONTINUE HERE .
    ...
  }
}
```

Please compile all the JAVA classes and run the solver.
Please print out the solution from the leaves of the elimination tree.