**ALGORITHM FOR COMPLETION TIME**

1. For each node, do step 1.1 (until completion times of all nodes are calculated)
   1. If the predecessors are completed, then take the latest completions  
      time of the predecessors and add required time for this node.
2. The node with the latest completion time determines the earliest completion time for project.

**ALGORITHMFOR MARKING CRITICAL PATH**

1. Start with the node(s) with the latest completion time(s); mark it (them) as  
   critical.
2. Select the predecessor(s) of the critical node(s) with latest completion  
   time(s); mark it (them) as critical. Continue Step 2 until reaching the  
   starting node(s).

**Example:** For the below stated activity chart:

Input:-

|  |  |  |
| --- | --- | --- |
| Activity | Intermediate predecessor | Duration (in weeks) |
| A | - | 8 |
| B | - | 10 |
| C | A,B | 8 |
| D | A | 9 |
| E | B | 5 |
| F | C,D | 3 |
| G | D | 2 |
| H | F,G | 4 |
| I | E,F | 3 |

The slack values are as follows

Output:-

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | Start Time | Completion Time | Critical Path |
| A | 0 | 8 |  |
| B | 0 | 10 | \* |
| C | 10 | 18 | \* |
| D | 8 | 17 |  |
| E | 10 | 15 |  |
| F | 18 | 21 | \* |
| G | 17 | 19 |  |
| H | 21 | 25 | \* |
| I | 21 | 24 |  |

The critical path is B->C->F->H.

**Problem:**

Now, write a java code which will take file as input (contains activity matrix) and gives us the start time, completion time for each activity. Also find the Critical path for the given test case.

**Note**:-

Input and output format must be same as given example.