CPM Processing Chart Output

- The code from the main method was used to construct the following critical path method table
- See the homework for a graphical representation

```
Thomass-MacBook-Pro:cpm thomaspang$ javac *.java
Thomass-MacBook-Pro:cpm thomaspang$ java HomeworkDriver
-----------------------------------------------------------------
|  Task | Value |    ES |    EF |    LF |    LS | LS-ES | LF-EF |
-----------------------------------------------------------------
|     A |     8 |     0 |     8 |     8 |     0 |     0 |     0 |
-----------------------------------------------------------------
|     B |    60 |     8 |    68 |    68 |     8 |     0 |     0 |
-----------------------------------------------------------------
|     C |    15 |    78 |    93 |   108 |    93 |    15 |    15 |
-----------------------------------------------------------------
|     D |    60 |   108 |   168 |   168 |   108 |     0 |     0 |
-----------------------------------------------------------------
|     E |     9 |   168 |   177 |   177 |   168 |     0 |     0 |
-----------------------------------------------------------------
|     F |    38 |   177 |   215 |   215 |   177 |     0 |     0 |
-----------------------------------------------------------------
|     G |     4 |     8 |    12 |    36 |    32 |    24 |    24 |
-----------------------------------------------------------------
|     H |    10 |    68 |    78 |    78 |    68 |     0 |     0 |
-----------------------------------------------------------------
|     I |    30 |    78 |   108 |   108 |    78 |     0 |     0 |
-----------------------------------------------------------------
|     J |    42 |    12 |    54 |    78 |    36 |    24 |    24 |
-----------------------------------------------------------------
Thomass-MacBook-Pro:cpm thomaspang$
```

Main Method

```
public static void main(String[] args)
    {
        // Define task nodes presented in diagram
        TaskNode taskA = new TaskNode("A", 8);
        TaskNode taskB = new TaskNode("B", 60);
        TaskNode taskC = new TaskNode("C", 15);
        TaskNode taskD = new TaskNode("D", 60);
        TaskNode taskE = new TaskNode("E", 9);
        TaskNode taskF = new TaskNode("F", 38);
        TaskNode taskG = new TaskNode("G", 4);
        TaskNode taskH = new TaskNode("H", 10);
        TaskNode taskI = new TaskNode("I", 30);
        TaskNode taskJ = new TaskNode("J", 42);

        // Set directional edges
        taskA.addSuccessors(taskB, taskH, taskG);
        taskB.addSuccessors(taskC, taskH);
        taskC.addSuccessors(taskD, taskE);
        taskD.addSuccessors(taskE);
        taskE.addSuccessors(taskF);
        taskG.addSuccessors(taskH, taskI, taskJ);
        taskH.addSuccessors(taskC, taskD, taskI);
        taskI.addSuccessors(taskD, taskE);
        taskJ.addSuccessors(taskI);

        // Calculate the critical path and output table
        CriticalPath.calculate(taskA, taskF);
    }
```