A) The optimal substructure for this question is when we drop a sheet of glass from a random floor. There will be two possible outcomes, either the sheet breaks or it doesn’t. So, there are three base cases: 1) When floors = 0, no trials are required since there is no floor to test on. 2) When floors = 1, only one trial is required. 3) When there are one sheet and x floors, then the worst case amount of trial is x starting all the way from the bottom floor to the maximum x floors. If the first floor doesn’t break you go to the second and so on. For the rest of the case, if the sheet breaks after dropping from the x floor, then we would only need to check the floors lower than x with remaining sheets. The subproblem reduces to n-1 sheets and x-1 floors. If the sheet doesn’t break, then we only check the floors above x floor with original number of sheets. The subproblem reduces to n sheets and total amount of floors – x floor. By minimize the minimum amount of trials in the worst case, we take the maximum of the two cases and choose the floor with the minimum number of trials.

B)

```
                                   G(4,2)

           /                 |                   |               \

          /  \              /  \                /   \            /   \

     G(0,1)  G(3,2)    G(1,1) G(2,2)      G(2,1) G(1,2)    G(3,1) G(0,2)

          /      |      \          /   \

        /  \    /  \    /  \      /  \   /  \

  G(0,1) G(2,2)  G(1,1)G(1,2) G(2,1)G(0,2)  G(0,1)G(1,2)  G(1,1) G(0,2)

          /   \

        /  |    |  \

 G(0,1)G(1,2)  G(1,1) G(0,2)
```

D) 8 = G(0,1), G(0,2), G(1,1), G(1,2), G(2,1), G(2,2), G(3,1) G(3,2)

E) #subproblem = number of sheets * number of floors

F) For memorizing GlassFallingRecur, I added an array to store the value for each individual subproblems. So before going into recursion, if the value of trials has been store then just return the value. Or else goes into calculation and then store the suitable value in the array. So the PC won’t need to recalculate the same subproblems over and over again.

```
A)                              CR(5)
              /                   |             |             |             \
           CR(4)               CR(3)         CR(2)         CR(1)         CR(0)
          /  |   |   |        |   |   |       | |           |
     CR(3)CR(2)CR(1)CR(0)     CR(2)CR(1)CR(0)  CR(1)CR(0)     CR(0)
     |  |  |  | |     \       | |   |          |
  CR(2)CR(1)CR(0) CR(1)CR(0) CR(0)   CR(1) CR(0)  CR(0)   CR(0)
  | |    \     |                    |
CR(1)CR(0)  CR(0)   CR(0)          CR(0)
  |
CR(0)
```

B)

Length i	1	2	3	4	5	6
price pi	1	4	30	12	55	24
pi / i	1	2	10	3	11	4

If the length of the given rod is 6. From the greedy algorithm we cut the rod of length 5 since it has the maximum density with the price of 55. The rod is left with length 1 and with the total price would be 56. However, this isn't the optimal solution. The optimal way is have two rods length of 3, which has the price 60.