Pranjit Kumar Kalita
Project #4

Efficient Packing of Rectangles using Binary Trees					
File Name	Elapsed Time for Packing & Efficient Packing Operations	Width	Height	Best Width	Best Height
r0.po	0.000000e+00 / 0.000000e+00	1.100000e+01	1.000000e+01	1.100000e+01	8.000000e+00
r1.po	0.000000e+00 / 0.000000e+00	1.663549e+06	1.812356e+06	1.560376e+06	1.881875e+06
r2.po	0.000000e+00 / 0.000000e+00	3.411589e+06	4.923995e+06	2.622907e+06	4.923995e+06
r3.po	0.000000e+00 / 0.000000e+00	9.793506e+06	2.573651e+06	8.618902e+06	2.728840e+06
r4.po	0.000000e+00 / 0.000000e+00	9.653391e+06	1.018544e+07	9.515508e+06	1.018544e+07
r5.po	0.000000e+00 / 0.000000e+00	2.745456e+07	1.320092e+07	2.286623e+07	1.365289e+07
r6.po	0.000000e+00 / 0.000000e+00	1.100000e+01	1.500000e+01	1.300000e+01	1.100000e+01

Space Complexity of my Rerooting algorithm: **O(log(n))**, since it is determined by the height/depth of a strictly binary tree, which is log(n).
Space Complexity of my Tree creation algorithm: **O(log(n))**, since the maximum number of nodes passed into the stack that is used for creating the binary tree is log(n).

Time Complexity of my Rerooting algorithm: **O(n)**, since we are visiting each node once, and we are only calculating the result for each node based on pre-determined numbers.
Time Complexity of my Tree creation algorithm: **O(n)**, since we are creating for every node separately.