**// Complete and Add this utility function to HW4 for EC**

**// Climbs up from the vertex to compute Height and Balance Factor**

**// of all ancestors and displays them as they are computed.**

**// This should be called whenever a vertex (N) is inserted**

**// This should be called whenever a vertex is deleted (i.e. delete V)**

**void BST::climbup(Vertex\* V)**

**{**

**cout << "...Start climbing up to adjust heights ......" << endl;**

**while (V != NULL)**

**{**

**// \*\* compute V->Height based on the left/right children's heights**

**// \*\* compute V->Balance based on the left/right children's heights**

**cout << "..." << V->Elem << "'s height: "**

**<< V->Height << " with balance: "**

**<< V->Balance << endl;**

**// \*\* go up to the parent**

**}**

**}**

**Compile with hw4client.cpp and test it.**

**Run bstEC.out to check the correct output.**

**-- Rotation EC for Later Submission -------------**

**Note that rotated.out shows what happens if you rotate the tree**

**everytime Balance -2 or 2 is found, and the tree is displayed**

**as soon as the rotation is completed.**

**Hint: I called fixIt from climbUp whenever 2 or -2**

**was found as V->Balance.**

**And fixIt called the correct**

**rotation routine(s) based on V's BF and the BF of**

**a sub-tree.**

**And then Display in Inorder to show the fixed tree.**

**// Red is where +2 or -2 was found**

**void BST::fixIt(Vertex\* Red)**

**{ cout << ">>>Fixing an unbalanced node at " << Red->Elem << endl;**

**Vertex\* Pivot;**

**if (Red->Balance == 2 && Red->Right->Balance == 1) // case 1**

**{**

**cout << " case 1: right heavy so rotate to left" << endl;**

**Pivot = rotateLeft(Red);**

**}**

**// handle all other cases here**

**if (Root == Red)**

**{ Root = Pivot; // new Root is set**

**cout << " New root is: " << Root->Elem << endl;}**

**cout << ">>>Fixed" << endl;**

**Display();**

**}**

**rotated.out shows you the correct output.**