// Parse professor and student solutions.

// Collapse each and run both uncollapsed and collapsed trees through the algorithm

// Output higher grade (in case somehow collapsing yields the student a lower grade)

// So far thinking of 3 categories to consist of total grade

// 1. At least having required nodes and edges

// 2. Similarity %

// 3. Same outcomes with same inputs

Tree canonicalSolution

Tree studentSolution

Tree canonicalSolutionCollapsed

Tree studentSolutionCollapsed

Procedure Preprocessing:

Check validity of canonical XML solution

Parse canonical solution from XML into object

Check validity of student XML solution

Parse student solution from XML into object

Run collapsing algorithm on canonical solution

Save collapsed professor tree in new object

Run collapsing algorithm on student solution

Save collapsed student tree in new object

End Procedure

// Run the grading procedure on both the uncollapsed and collapsed versions

// Return the higher grade

Procedure Grading

// Baseline grading, ensure at least relevant nodes are present

Map correctNodes

Map correctEdges

Map studentNodes

Map studentEdges

FillNodes(correctNodes)

FillEdges(correctEdges)

FillNodes(studentNodes)

FillEdges(correctEdges)

// Check to see if student has at least all nodes and edges present

// If so assign a baseline grade like 50% of the credit

Check(correctNodes, studentNodes)

Check(correctEdges, studentEdges)

Add some portion of total grade

// Some portion of the credit is how similar the trees are

CheckSimilarity(canonicalSolution, studentSolution)

Add some portion of total grade

// Run sample cases to see if they arrive at the same result of close

// Som portion of the credit given here

RunSampleTests()

Add some portion of the total grade

return grade

End Procedure

Procedure FillNodes

If current node is not null

correctNodes.add(current.name, current.type)

For all children of current

FillNodes(children)

End for

End if

End Procedure

Procedure FillEdges

If current node is not null

correctEdges.add(edge, weight)

For all children of current

FillEdges(children)

End For

End If

End Procedure