Assignment 5

A. (a) Design a pseudo-code algorithm, sum(T), that sums the values in the internal nodes of a binary tree (see hint in the in-class exercise in the class notes).

Algorithm: sum(T)

return sumHelper(T, T.root()----------O(1)

Algorithm: sumHelper(T, P)

sum ← P.element()----------O(1)

if T.isExternal(P) then ----------O(1)

return sum----------O(1)

else

sum ←P.element()----------O(1)

leftSum ←sumHelper(T, T.leftChild(P)----------O(n)

rightSum ← sumHelper(T, T.rightCild(P)----------O(n)

return sum + leftSum + rightSum----------O(1)

the running time --------**O(n)**

(b) Using the Tree.js implementation of the BinaryTree ADT, implement in JavaScript the function, sum(T), that sums the values in a binary tree.

```
181
182  function sum(T) {
183      return sumHelper(T, T.root());
184  }
185
186  function sumHelper(T, p) {
187      if (T.isExternal(p)) {
188          return 0;
189      } else {
190          let sum = p.element();
191          let lsum = sumHelper(T, T.leftChild(p));
192          let rsum = sumHelper(T, T.rightChild(p));
193          return lsum + rsum + sum;
194      }
195  }
196
197  console.log(sum(t0))
```

B. (a) Design a pseudo-code algorithm, findMax(T), that finds the maximum value stored in a binary tree.

Algorithm: findMaximum(T)

return findMaximumHelper(T, T.root())----------O(1)

Algorithm: findMaximumHelper(T,p)
if T.isExternal(p) then ----------O(1)
return -∞ ----------O(1)
else
leftMax ← findMaximumHelper(T, T.leftChild(p))----------O(n)
rightMax ← findMaximumHelper(T, T.rightChild(p))----------O(n)

return max(leftMax, rightMax, p.element()) ----------O(1)

so the running time is ---------**O(n)**

(b) Based on the Tree.js implementation of the binary tree, implement in JavaScript the function, findMax(T), that finds the maximum in a tree.

```
198 function findMaximum(T){
199     return findMaximumHelper(T, T.root())
200 }
201 function findMaximumHelper(T, p){
202     if(T.isExternal(p)){
203         return -Infinity;
204     }else{
205         let leftMax = findMaximumHelper(T, T.leftChild(p))
206         let rightMax = findMaximumHelper(T, T.rightChild(p))
207         return Math.max(leftMax, rightMax, p.element())
208     }
209 }
210 console.log(findMaximum(t0))
```

C. (a) Based on the EulerTour template class provided in Tree.js, implement a function sum that sums the elements in a binary tree. This is done by creating a subclass of EulerTour that overrides one or more hook methods in the superclass.

(b) Based on the EulerTour, implement a function the finds the maximum value in the tree.