MAHARASHI INTERNATIONAL UNIVERSITY

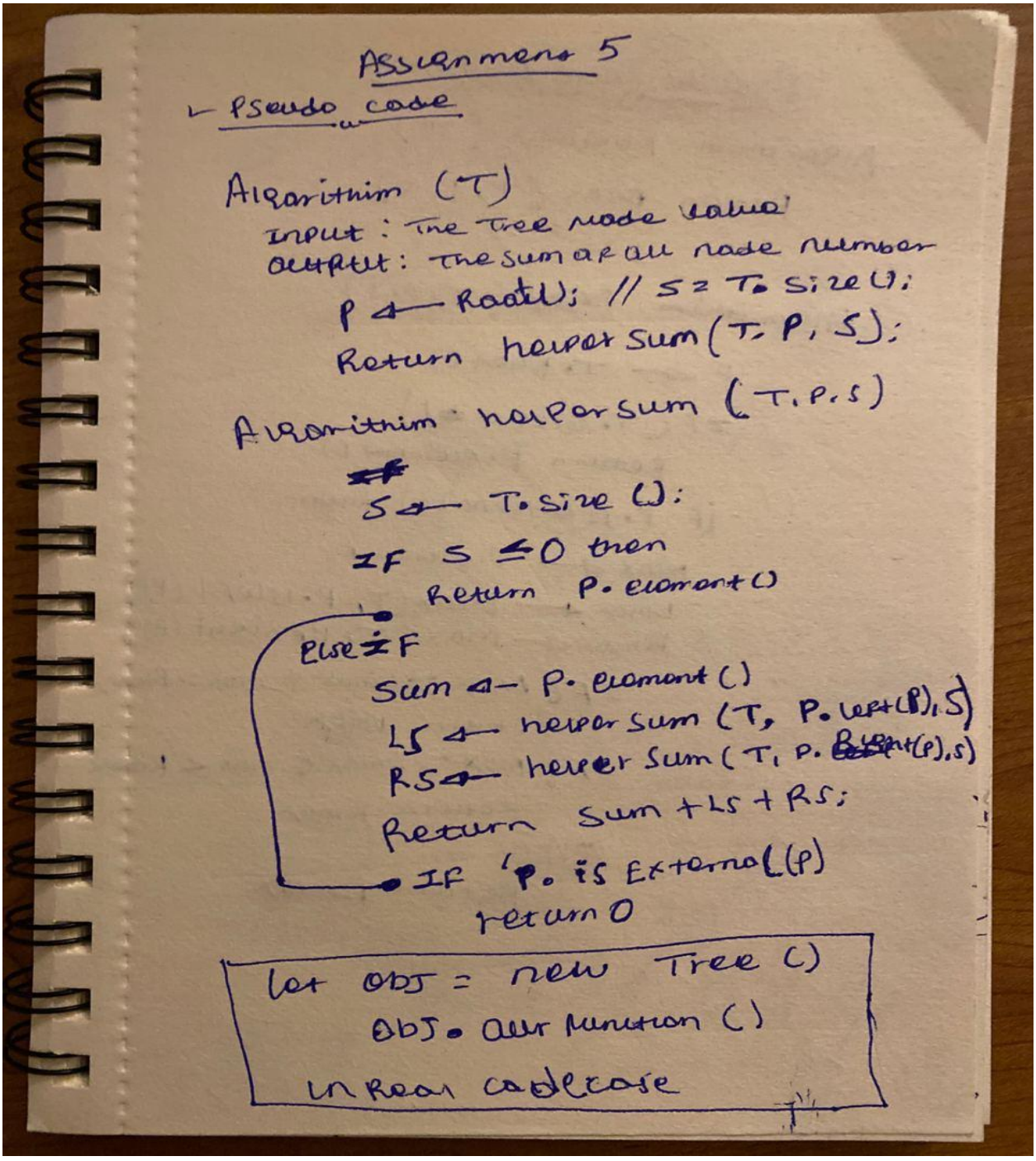
Assignment Five

NOVEMBER 2, 2020

KASAHUN TEHONE
IDNO: 111705

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).

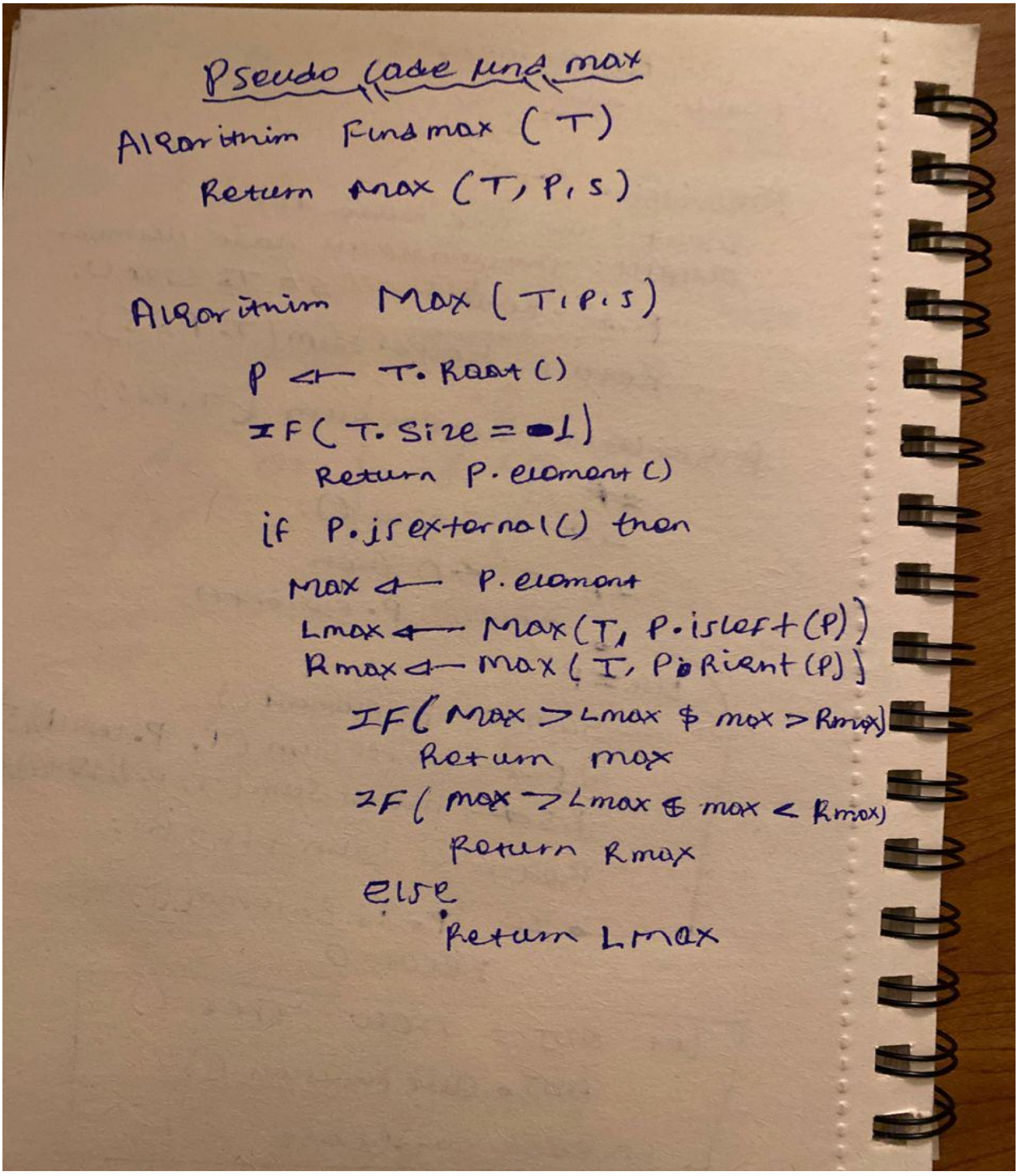


(b) Using the Tree.js implementation of the BinaryTree ADT, implement in JavaScript the function, sum(T), that sums the values in a binary tree.

```
{
function sum(t){
    return sumHelper(t,t.root());
}

function sumHelper(t,p){
    //  let sum=p

    if(t.isExternal(p)){
        return 0;
    }
    else{
        sum=p.element()
        let left=sumHelper(t,t.leftChild(p));
        let right=sumHelper(t,t.rightChild(p));
        //return sum + left.element() + right.element();
        return sum + left + right;
    }

}

let t0 = new BinaryTree();

let printer = new Print();

printer.print(t0);

let r = t0.insertRoot(300);
printer.print(t0);

let l1 = t0.insertLeft(r, 200);
let r1 = t0.insertRight(r, 400);
printer.print(t0);
t0.insertRight(l1, 250);
l1 = t0.insertLeft(l1, 100);
t0.insertRight(l1, 150);
l1 = t0.insertLeft(l1, 50);
l1 = t0.insertLeft(r1, 350);
r1 = t0.insertRight(r1, 500);
t0.insertLeft(r1, 450);
r1 = t0.insertRight(r1, 600);
```

```
t0.insertLeft(r1, 550);
r1 = t0.insertRight(r1, 800);
printer.print(t0);
t0.insertLeft(r1, 700);
printer.print(t0);


console.log("the sum is ",sum(t0));
}
```

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



(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.

```
function max(T)
```

```
{
    return maxHelper(T,T.root())
}
function maxHelper(T,p){
    if(T.isExternal(p)){
        return "the max is :",p.element();
    }

    else {
        let max=p.element();
        let lmax = maxHelper(T, T.leftChild(p));
        let rmax = maxHelper(T, T.rightChild(p))

        return Math.max(lmax, rmax, max)
    }
}
```

B.(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.

```
class Sum extends eulerTour {
    visitExternal(T, p, r){
        r[1] = 0;
    }

    visitPostOrder(T, p, r){

        r[1] = r[0] + r[2] + p.element();
    }

    sum(T){

        return eulerTour(T, T.root())
    }
}

function sumHelper1(T, p){
    if(T.isExternal(p)){
        return 0;
    }
```

```
    else {
       let lsum = sumHelper1(T, T.leftChild(p))
       let rsum = sumHelper1(T, T.rightChild(p))

       return lsum + rsum + p.element();
    }
}
```

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

```
class Max extends eulerTour {
    visitExternal(T, p, r){
        r[1] = -Infinity
    }

    visitPostOrder(T, p, r){
        r[1] = Math.max(r[0], r[2], p.element())
    }
    max(T){
        return eulerTour(T, T.root())
    }
}

function maxHelper(T){
    if(T.isExternal(p)){
        return -Infinity
    }

    else {
        lmax = maxHelper(T, T.leftChild(p));
        rmax = maxHelper(T, T.rightChild(p))

        return Math.max(lmax, rmax, p.element())
    }
}
```