CSE 326: Data Structures
Deletion of the Minimum Key from an AVL Tree
October 22, 1998

function AVLTreeDeleteMin(**locative** T) **returns** (**key, info, boolean**)
comment: In the tree pointed to by T, deletes the minimum key and returns that record's Key and Info fields, and also returns true if and only if the deletion caused the height of the tree to decrease;

```
begin
    declare K: key;
    declare I: info;
    declare decreased: boolean;
    declare B: integer;
    if T = Λ then error;
    if LC(T) = Λ
        then begin
            K ← Key(T);
            I ← Info(T);
            T ⇐ RC(T);
            return (K, I, true)
        end
        else begin
            (K, I, decreased) ← AVLTreeDeleteMin(LC(T));
            if not decreased
                then return (K, I, false);
                else case Balance(T) of
                    −1: Balance(T) ← 0;
                        return (K, I, true);
                     0: Balance(T) ← +1;
                        return (K, I, false);
                    +1: case Balance(RC(T)) of
                            +1: Rotate(T, −1);
                                Balance(T) ← 0;
                                Balance(LC(T)) ← 0;
                                return (K, I, true);
                             0: Rotate(T, −1);
                                Balance(T) ← −1;
                                Balance(LC(T)) ← +1;
                                return (K, I, false);
                            −1: B ← Balance(LC(RC(T)));
                                Rotate(RC(T), +1);
                                Rotate(T, −1);
                                Balance(T) ← 0;
                                if B = +1
                                    then Balance(LC(T)) ← −1
                                    else Balance(LC(T)) ← 0;
                                if B = −1
                                    then Balance(RC(T)) ← +1
                                    else Balance(RC(T)) ← 0;
                                return (K, I, true);
                        end case
                end case
        end
end .
```