DSA LAB
Lab Assignment number 14

Name: Aamir Ansari **Batch:** A **Roll no:** 01

Aim: To implement various operations on AVL tree

Theory:

Algorithm to insert a node in AVL tree:

insertNode (ROOT, DATA)

Step 1: IF ROOT = NULL, then
Allocate memory for newNode
newNode->DATA=DATA
newNode->LEFT=newNode->RIGHT=NULL
newNode->HEIGHT=1
RETURN newNode;
[END OF IF]
Step 2: IF DATA < ROOT->DATA
ROOT->LEFT=insertNode(ROOT->LEFT, DATA)
ELSE IF DATA > ROOT->DATA
ROOT->RIGHT=insertNode(ROOT->RIGHT,DATA)
ELSE
RETURN ROOT
[END OF IF]
Step 3: ROOT->HEIGHT = 1 + MAX ((height(ROOT->LEFT),height(ROOT->RIGHT))
Step 4: SET BALANCE = ROOT->LEFT->HEIGHT – ROOT->RIGHT->HEIGHT
Step 5: IF BALANCE>1 AND DATALEFT->DATA
RETURN rotateRight(ROOT)
IF BALANCE<-1 AND DATA>ROOT->RIGHT->DATA
RETURN rotateLeft(ROOT)
IF BALANCE>1 AND DATA>ROOT->LEFT->DATA
ROOT->LEFT=rotateLeft(ROOT->LEFT)
RETURN rotateRight(ROOT)
IF BALANCE<-1 AND DATA<ROOT->RIGHT->DATA
ROOT->RIGHT=rotateRight(ROOT->RIGHT)
RETURN rotateLeft(ROOT)
[END OF IF]
Step 6: RETURN ROOT

Algorithm to delete a node in AVL tree:

deleteNode (ROOT, DATA)
Step 1: IF ROOT = NULL, then
RETURN ROOT

```
        [END OF IF]
Step 2: IF DATA < ROOT->DATA
                ROOT->LEFT=deleteNode(ROOT->LEFT, DATA)
        ELSE IF DATA>ROOT->DATA
                ROOT->RIGHT=deleteNode(ROOT->RIGHT,DATA)
        ELSE
                IF ROOT->LEFT=NULL OR ROOT->RIGHT=NULL
                        SET TEMP=ROOT->LEFT ? ROOT->LEFT : ROOT->RIGHT
                        IF TEMP=NULL
                                TEMP=ROOT
                                ROOT=NULL
                        ELSE
                                ROOT=TEMP
                        [END OF IF]
                        FREE(TEMP)
                ELSE
                        TEMP=smallestNode(ROOT->RIGHT)
                        ROOT->DATA=TEMP->DATA
                        ROOT->RIGHT=deleteNode(ROOT->RIGHT, TEMP->DATA)
                [END OF IF]
        [END OF IF]
Step 3: IF (ROOT=NULL)
                RETURN ROOT
Step 4: ROOT->HEIGHT = 1 + MAX(height(ROOT->LEFT),height(ROOT->RIGHT))
Step 5: SET BALANCE = ROOT->LEFT->HEIGHT – ROOT->RIGHT->HEIGHT
Step 6: IF BALANCE>1 AND DATA < ROOT->LEFT->DATA
                RETURN rotateRight(ROOT)
        IF BALANCE<-1 AND DATA>ROOT->RIGHT->DATA
                RETURN rotateLeft(ROOT)
        IF BALANCE>1 AND DATA>ROOT->LEFT->DATA
                ROOT->LEFT=rotateLeft(ROOT->LEFT)
                RETURN rotateRight(ROOT)
        IF BALANCE<-1 AND DATARIGHT->DATA
                ROOT->RIGHT=rotateRight(ROOT->RIGHT)
                RETURN rotateLeft(ROOT)
        [END OF IF]
Step 7: RETURN ROOT
```

Algorithm to search an element in AVL tree:

```
search (ROOT, VAL)
Step 1: IF ROOT ->DATA = VAL OR ROOT = NULL, then
                Return ROOT
        ELSE
                IF VAL < ROOT ->DATA
                        Return search(ROOT->LEFT,VAL)
```

```
            ELSE
                Return search(ROOT->RIGHT,VAL)
            [END OF IF]
        [END OF IF]
Step 2: EXIT
```

Algorithm to find height/depth of AVL tree:

```
Height (ROOT)
Step 1: IF ROOT = NULL, then
            Return 0
        ELSE
            Return ROOT->HEIGHT
        [END OF IF]
Step 2: EXIT
```

Algorithm to count total number of nodes in AVL tree:

```
totalNodes (ROOT)
Step 1: IF ROOT = NULL, then
            Return 0
        ELSE
            Return totalNodes(ROOT ->LEFT) + totalNodes(ROOT ->RIGHT) + 1
        [END OF IF]
Step 2: EXIT
```

Algorithm to display the AVL tree:

```
displayTree (ROOT, space)
Step 1: IF (ROOT=NULL) then
            Goto step 8
        [END OF IF]
Step 2: SET space = space + 8
Step 3: displayTree (ROOT->RIGHT, space)
Step 4: SET I=1
Step 5: Repeat while I < space
            PRINT " "
            I=I+1
        [END OF LOOP]
Step 6: PRINT ROOT->DATA
Step 7: displayTree (ROOT->LEFT, space)
Step 8: EXIT
```