Name : Bryan Orville Audric  
NIM: 2602160750

struct node

int key

struct node \*left

struct node \*right

int height

END struct

Int Count = 0

int Height(struct node \*NodeIn):

if NodeIn is NULL

return 0

else

return NodeIn->height

END IF

END Function

int Max (int a, int b):

if a>b

return a

else

return b

END IF

END Function

int CheckBalance(struct node \*NodeIn):

if NodeIn is NULL

return 0

else

return Height(NodeIn->left)-Height(NodeIn->right)

END IF

END Function

struct node \*RightRotate(struct node \*Current):

struct node \*NewCurrent = Current->left

struct node \*RightNewCurrent = NewCurrent->right

NewCurrent->right=Current

Current->left=RightNewCurrent

Current->height = 1 + Max(Height(Current->left),Height(Current->right))

NewCurrent->height = 1 + Max(Height(NewCurrent->left),Height(NewCurrent->right))

return NewCurrent

END Function

struct node \*LeftRotate(struct node \*Current):

struct node \*NewCurrent = Current->right

struct node \*LeftNewCurrent = NewCurrent->left

NewCurrent->left=Current

Current->right=LeftNewCurrent

Current->height = 1 + Max(Height(Current->left),Height(Current->right))

NewCurrent->height = 1 + Max(Height(NewCurrent->left),Height(NewCurrent->right))

return NewCurrent

END Function

struct node \*insert(struct node \*root, int KeyInput):

if root is NULL

struct node \*NewNode = (struct node\*)malloc(sizeof(struct node))

NewNode->key=KeyInput

NewNode->left=NULL

NewNode->right=NULL

NewNode->height=1

return NewNode

END IF

If KeyInput<root->key

root->left=insert(root->left,KeyInput)

else if(KeyInput>root->key)

root->right=insert(root->right,KeyInput)

else

return root

END IF

root->height = 1 + Max(Height(root->left),Height(root->right))

int BalanceFactor = CheckBalance(root)

if BalanceFactor>1 && KeyInput<root->left->key

return RightRotate(root)

END IF

If BalanceFactor>1 && KeyInput>root->left->key

root->left = LeftRotate(root->left)

return RightRotate(root)

END IF

If BalanceFactor<-1 && KeyInput>root->right->key

return LeftRotate(root)

END IF

If BalanceFactor<-1 && KeyInput<root->right->key

root->right = RightRotate(root->right)

return LeftRotate(root)

END IF

return root

END Function

struct node \*MinVal(struct node \*root):

struct node \*Current=root

while Current->left is not NULL

Current=Current->left

END while

return Current

END Function

struct node \*Deletenode (struct node \*root, int KeyIn):

if root is NULL

return root

END IF

If KeyIn<root->key

root->left=Deletenode(root->left,KeyIn)

else if KeyIn>root->key

root->right=Deletenode(root->right,KeyIn)

else

Count ++;

If Count is 1

Display”Data found\n”

Display”Value %d was deleted\n”,KeyIn

END IF

If root->left is NULL

struct node \*temp = root->right

free(root)

return temp

else if root->right is NULL

struct node \*temp = root->left

free(root)

return temp

else

struct node \*temp = MinVal(root->right)

root->key=temp->key

root->right=Deletenode(root->right,temp->key)

END IF

END IF

root->height = 1 + Max(Height(root->left),Height(root->right))

int BalanceFactor = CheckBalance(root)

if BalanceFactor>1 && CheckBalance(root->left)>=0

return RightRotate(root)

END IF

If BalanceFactor>1 && CheckBalance(root->right)<0

root->left=LeftRotate(root->left)

return RightRotate(root)

END IF

If BalanceFactor<-1 && CheckBalance(root->right)<=0

return LeftRotate(root)

END IF

If BalanceFactor<-1 && CheckBalance(root->left)>0

root->right=RightRotate(root->right)

return LeftRotate(root)

END IF

return root

END Function

void PrintPreOrder(struct node \*root):

if root is not NULL

Display"%d ",root->key

PrintPreOrder(root->left)

PrintPreOrder(root->right)

END IF

END Function

void PrintInOrder(struct node \*root):

if root is not NULL

PrintInOrder(root->left)

Display "%d ",root->key

PrintInOrder(root->right)

END IF

END Function

void PrintPostOrder(struct node \*root):

if root is not NULL

PrintPostOrder(root->left)

PrintPostOrder(root->right)

Display "%d ",root->key

END IF

END Function

int main():

struct node \*root=NULL

int menu,Data,DeleteData

do

Display "1. Insertion\n"

Display "2. Deletion\n"

Display "3. Traversal\n"

Display "4. Exit\n”

Display "Choose : "

Input menu

switch(menu)

case 1:

Display "Insert: "

Input Data

root = insert(root,Data)

system("pause")

system("cls")

break

case 2:

Display "Delete : "

Input DeleteData

Count = 0

root = Deletenode(root,DeleteData)

system("pause")

system("cls")

break

case 3:

Display "Preorder: "

PrintPreOrder(root)

Display "\n"

Display "Inorder: "

PrintInOrder(root)

Display "\n"

Display "Postorder: "

PrintPostOrder(root)

Display "\n"

system("pause")

system("cls")

break

END Switch

While menu is not 4

END DO While

Display "Thank you\n"

END Function