BST binary search tree

Construct a BST by inserting name of month from JAN to DEC

JAN

FEB

MAR

APR

MAY

JUN

JUL

AUG

SEP

OCT

NOV

DEC

//every time we decide to insert we make it faster by assign number based on their alphabetical order

JAN 5

FEB 4

MAR 8

APR 1

MAY 9

JUN 7

JUL 6

AUG 2

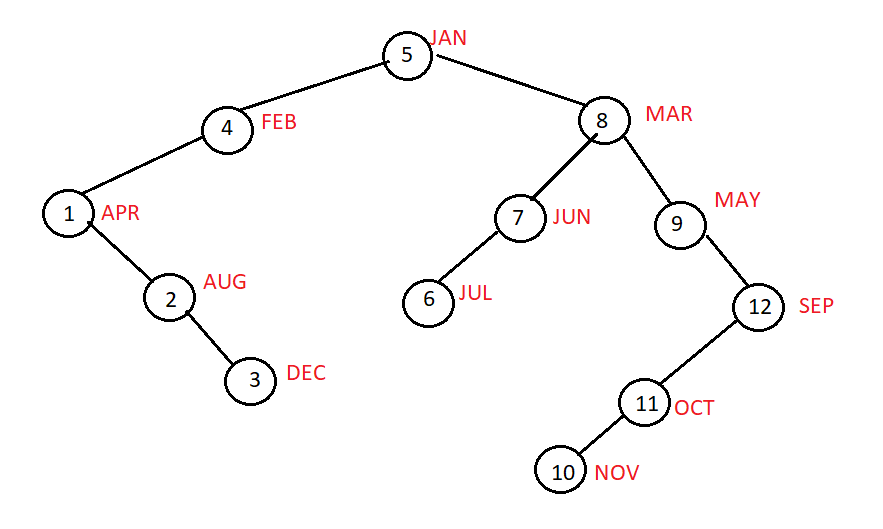
SEP 12

OCT 11

NOV 10

DEC 3

Root of the tree is January so



display in order traversal will display in alphabetical order

void displayInOrder(node \*r)

{

if(r!=NULL)

{

displayInOrder(r->left); //visiting LC

cout<<”r->info”<<” “; //visit P

displayInOrder(r->right); //visit RC

}//endif

}//end dispalyInOrder

//=------------------------

another example

we want to see only the leaves of the tree

ones that don’t have children

we want to display data of items without children:

void displayInOrder(node \*r)

{

if(r!=NULL)

{

displayInOrder(r->left); //visiting LC

if(r->left==NULL || r->right==NULL)//then print content of node

{cout<<r->info;}

displayInOrder(r->right); //visit RC

}//endif

}//end dispalyInOrder

//collect total numbers lets say we had ints in the tree

int total=0; //declare total somewhere else not inside the recursive

int displayInOrder(node \*r, int &total)//total is reference so value changes

// outside of this function

{

//DON’T INITIALIZE TOTAL IN RECURSIVE FUNCTION

if(r!=NULL)

{

displayInOrder(r->left); //visiting LC

total+=p->info;

displayInOrder(r->right); //visit RC

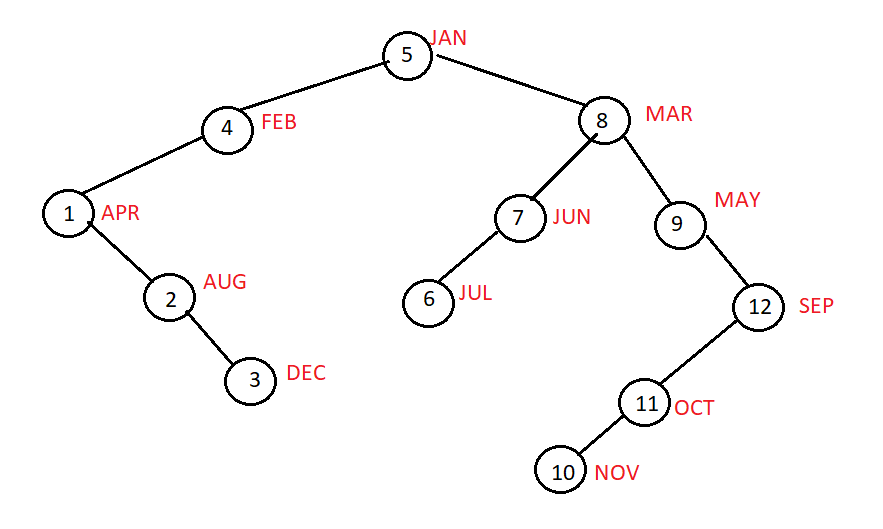
}//endif

}//end dispalyInOrder

//-------------------------------------

trace the following function and show its final output

calling statement: //using this tree but content are numbers



cout<<f(root); //means were pointing to 5 or the root

int f(node \*r) //we receive the root

{

if (r==NULL) return 0;

else return r->info+f(r->left)+f(r->right);

}

