truelines

DATE : __/__/____

Name: Manthan. M. Sonawane
Panel: E Roll No: 17
ADS

Assignment 2

- Title : Binary Tree and its Traversal

- Problem Statement : Implement binary tree using c++ and perform following operations : creation of binary tree and traversal (recursive or non-recursive)

- Objective :
- To study data structure : tree and binary tree
- To study different traversal in Binary tree.
- To study recursive and Non recursive approach of programming

- Theory

A tree is a data structure that represents a hierarchial structure and is used to store and organize data, it consist of nodes connected by edges, with one node designated as root and current as child nodes

- Definations

a) Binary Tree :

A binary tree is a data structure in which each node has atmost two children.

trueline
DATE : __/__/____

b) Full binary tree :
A binary tree with every node has either 0 or 2 children is called full binary tree.

c) Compute binary tree :
It's a binary tree in which every level is completely filled except for root level which is filled from left to right

d) Perfect binary tree :
A perfect binary tree is a tree in which all left nodes are at same level and every parent has two children.

e) Balanced binary tree :- In this the difference between the height of the left and right subtree of any nodes is less than or equal to one

f) Leaf node - node in a tree with no children

g) parent node : node in a tree that has atleast one child node

- Different traversal :

a) Inorder Traversal :
In an inorder traversal, the left subtree then the root node, and finally subtree.

b) Preorder Traversal :
In a preorder traversal the root node is visited first then the left subtree

truelines
DATE : __/__/____

Implementation:
- 64 bit open source linux or its derivatives
- Open source c++ programming tools like g++ / Eclipse Editor

Test conditions:

Inorder

```
inorder () {
    temp = root;
    while (1) {
      while (temp != Null)
    }
     push temp onto stack;
      temp = temp -> left;
    }
    if stack empty
     break;
    pop stack into temp;
       visit temp
    temp = temp -> right;
  }
}
```

Preorder

```
preorder () {
 temp = root;
while (1) {
  while (temp is not NULL)
 {
  visit temp;
  push temp onto stack;
```

truelines
DATE : __/__/____

```
        temp= temp -> left ;
    }
    if stack empty
    break;
    pop stack into temp;
    temp= temp -> right;
  }
}
```

Postorder

```
postorder_nr() {
      temp = root;
    while(1) {
      while (temp is not NULL)
    }
     push temp onto stack;
      temp = temp -> left;
    }
    if stack top right is NULL
    {
      pop stack into temp
      visit temp
    }
  while (stack not empty or stack -> right is
       temp)
  {
    pop stack into temp;
     visit temp;
  }
```

```
    if stack empty
      break;
    move temp to stack to right;
  }
}
```

Time Complexity

Creating a binary tree = O(n)
Inorder = O(n)
preorder = O(n)
postorder = O(n)

Conclusion:

Thus implemented different operation on CLL.

FAQ's

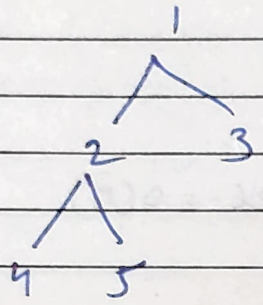
1) Explain any one application of binary tree with suitable example?

→ One common example of binary tree is for searching and sorting data efficiently.
For example, consider a dictionary of english words a binary search tree can be used to store the ~~cloud~~ words in such a way that searching can be done in logarithmic times each other.

2) Explain sequential Representation of binary tree with example?

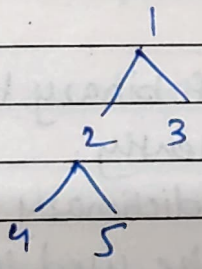
→ A sequential representation of binary tree is a way of storing the tree in an array where nodes are stored in specific order there are two ways:-

1) Breath First (level order) representation: In this representation the nodes of binary tree are stored level by level, from left to right.



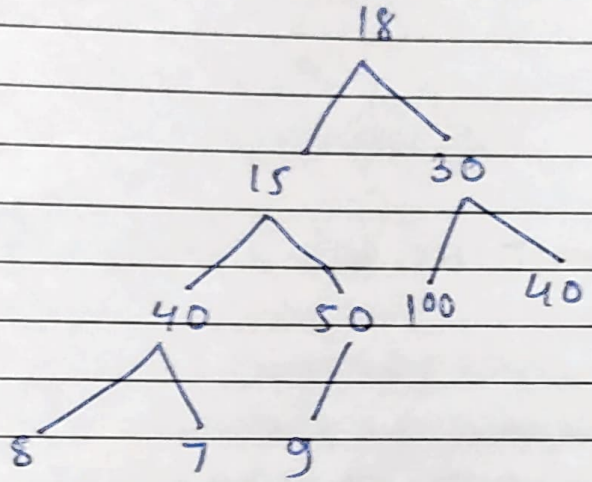
First representation of this tree would be : [1, 2, 3, 4, 5]

2) Depth First (preorder) representation:
In this representation the node are binary tree are stored in preorder manner root is stored first followed by left subtree and then right subtree



The depth first representation on this tree would be : [1, 2, 4, 5, 3]

3) Write inorder, preorder and postorder for following tree.



Inorder : 8, 40, 7, 15, 50, 9, 18, 100, 30, 40.
Preorder : 18, 15, 40, 8, 7, 50, 9, 30, 100, 40.
Postorder : 8, 7, 40, 9, 50, 15, 100, 40, 30, 18.

21/2/23

CODE

```
#include <iostream>
using namespace std;
class treenode{
   char data[10];
   treenode *left;
   treenode *right;
   friend class tree;
};
class tree{
   treenode *root;
public:
   tree();
   void create_nr();
   void inorder_r();
   void inorder_r(treenode*);
   void preorder_r();
   void preorder_r(treenode*);
   void postorder_r();
   void postorder_r(treenode*);
   void inorder_nr();
   void inorder_nr(treenode*);
   void preorder_nr();
   void preorder_nr(treenode*);
   void postorder_nr();
   void postorder_nr(treenode*);

};
class stack{
   int top;
   treenode *data[30];
   public:
   stack(){
       top=-1;
   }
   void push(treenode *temp);
   treenode *pop();

   bool empty();
   friend class tree;
};
bool stack::empty(){
```

CODE

```
   if(top==-1){
       return true;

   }
   else{
       return false;
   }
}
void stack:: push(treenode *temp){
   top++;
   data[top]=temp;
}
treenode *stack ::pop(){
   treenode *temp;
   temp =data[top];
   top--;
   return temp;
}
tree:: tree()
{
   root=NULL;
}
void tree:: inorder_nr(){
   stack s;
   treenode* temp;
   temp = root;
   while(1){
       while(temp!=NULL){
           s.push(temp);
           temp = temp->left;
       }
       if(s.empty()){
           break;
       }
       temp=s.pop();
       cout<<temp->data;
       temp = temp->right;
   }
}
void tree:: postorder_nr(){
   stack s;
   treenode* temp;
```

CODE

```
   temp = root;
   while(1){
       while(temp!=NULL){
           s.push(temp);
           temp = temp->left;
       }
       if(s.empty()){
           break;
       }
       temp=s.pop();
       temp = temp->right;
       cout<<temp->data;
   }
   while(s.empty() && s.top==-1){
       temp=s.pop();
       temp = temp->right;
       cout<<temp->data;



   }
}void tree:: preorder_nr(){
   stack s;
   treenode* temp;
   temp = root;
   while(1){
       while(temp!=NULL){
           s.push(temp);
           cout<<temp->data;
           temp = temp->left;
       }
       if(s.empty()){
           break;
       }
       temp=s.pop();
       temp = temp->right;
   }
}
void tree:: create_nr()
{
if (root==NULL)
{
    root= new treenode;
```

CODE

```
   cout<<"Enter root node:"<<endl;
   cin>>root->data;
   root->left=NULL;
   root->right=NULL;
   treenode *temp,*curr;
   char op;
do{
    temp=root;
   int flag=0;
   curr= new treenode;
       cout<<"Enter curr node:"<<endl;
       cin>>curr->data;
       curr->left=NULL;
       curr->right=NULL;
  while(flag==0)
  {
      char ch;
      cout<<"You want to add to left or right of "<<temp->data<<endl;
      cout<<"Enter l for left and r for right:"<<endl;
      cin>>ch;
      if (ch=='l')
      {
          if(temp->left ==NULL)
          {
              temp->left=curr;
              flag=1;
          }
          temp=temp->left;
      }
      else
      {
          if (ch=='r')
          {
              if (temp->right==NULL)
              {
                  temp->right=curr;
                  flag=1;
              }
              temp=temp->right;
          }
      }
  }
```

CODE

```
   cout<<"Do you want to continue: y or n -> ";
       cin>>op;
 }while(op=='y');
}
}
void tree ::inorder_r()
{
   inorder_r(root);
}
void tree:: inorder_r(treenode*temp){
   if(temp!=NULL){
        inorder_r(temp->left);
        cout<< temp->data;
       inorder_r(temp->right);
   }
}
void tree ::preorder_r()
{
   preorder_r(root);
}
void tree:: preorder_r(treenode*temp){
   if(temp!=NULL){
        cout<< temp->data;
        preorder_r(temp->left);
       preorder_r(temp->right);
   }
}
void tree ::postorder_r()
{
   postorder_r(root);
}
void tree:: postorder_r(treenode*temp){
   if(temp!=NULL){
        postorder_r(temp->left);
        postorder_r(temp->right);
        cout<< temp->data;
   }
}
int main() {

   tree a;
   a.create_nr();
```

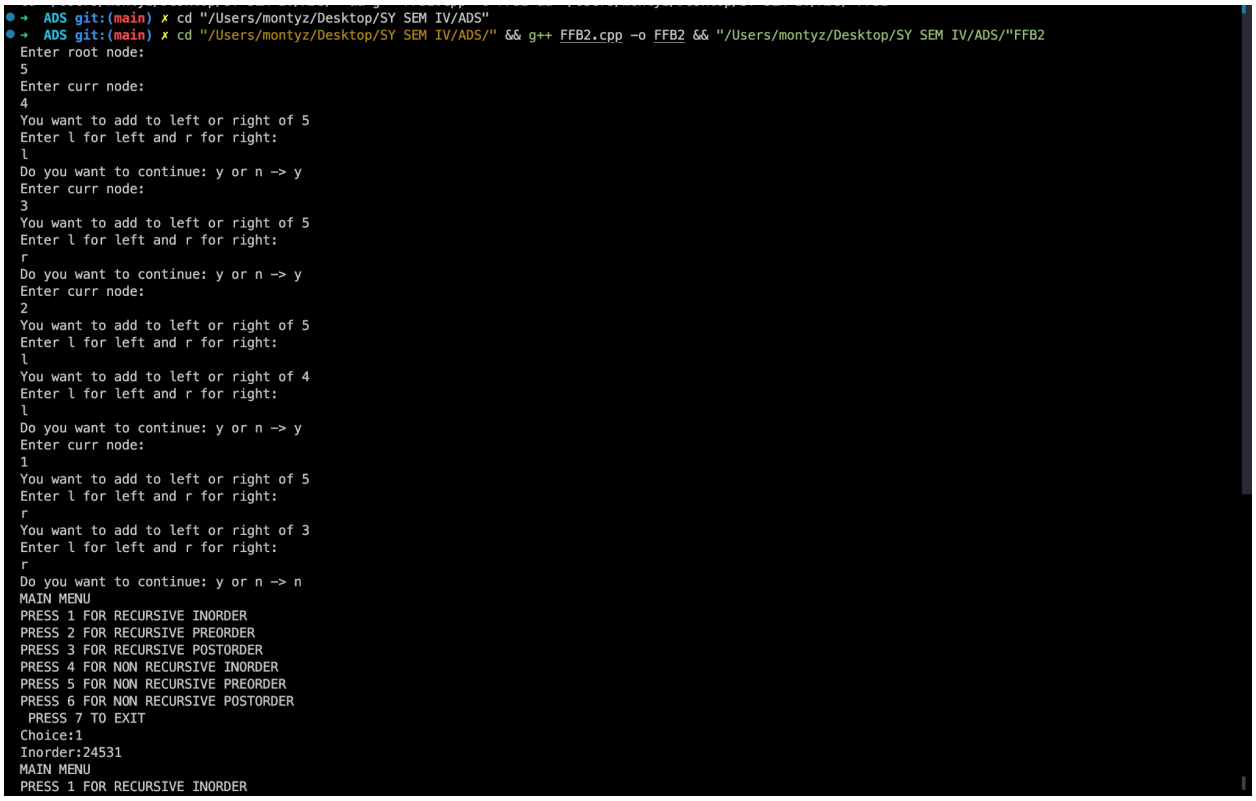
CODE

```
   int choice;

   do{
   cout<<"MAIN MENU \nPRESS 1 FOR RECURSIVE INORDER \nPRESS 2 FOR RECURSIVE PREORDER
\nPRESS 3 FOR RECURSIVE POSTORDER \nPRESS 4 FOR NON RECURSIVE INORDER \nPRESS 5 FOR
NON RECURSIVE PREORDER \nPRESS 6 FOR NON RECURSIVE POSTORDER \n PRESS 7 TO
EXIT"<<endl;
   cout<<"Choice:";
   cin>>choice;

switch(choice)
{
case 1:
   cout<<"Inorder:";
   a.inorder_r();
   cout<<endl;
   break;
case 2:
   cout<<"Preorder:";
   a.preorder_r();
   cout<<endl;
   break;
case 3:
   cout<<"Postorder:";
   a.postorder_r();
   cout<<endl;
   break;
case 4:
cout<<"Non recursive Inorder: ";
   a.inorder_nr();
   cout<<endl;
   break;

case 5:
cout<<"Non recursive Preorder: ";
   a.preorder_nr();
   cout<<endl;
   break;
case 6:
   cout<<"Non recursive Postorder: ";
   a.postorder_nr();
   cout<<endl;
```

CODE

```
   break;
case 7:
break;
default:
   cout<<"ENTER VALID CHOICE"<<endl;
break;
}
}while(choice!=7);


return 0;
}
```

OUTPUT



CODE


PRESS 2 FOR RECURSIVE PREORDER
PRESS 3 FOR RECURSIVE POSTORDER
PRESS 4 FOR NON RECURSIVE INORDER
PRESS 5 FOR NON RECURSIVE PREORDER
PRESS 6 FOR NON RECURSIVE POSTORDER
PRESS 7 TO EXIT
Choice:2
Preorder:54231
MAIN MENU
PRESS 1 FOR RECURSIVE INORDER
PRESS 2 FOR RECURSIVE PREORDER
PRESS 3 FOR RECURSIVE POSTORDER
PRESS 4 FOR NON RECURSIVE INORDER
PRESS 5 FOR NON RECURSIVE PREORDER
PRESS 6 FOR NON RECURSIVE POSTORDER
PRESS 7 TO EXIT
Choice:3
Postorder:24135
MAIN MENU
PRESS 1 FOR RECURSIVE INORDER
PRESS 2 FOR RECURSIVE PREORDER
PRESS 3 FOR RECURSIVE POSTORDER
PRESS 4 FOR NON RECURSIVE INORDER
PRESS 5 FOR NON RECURSIVE PREORDER
PRESS 6 FOR NON RECURSIVE POSTORDER
PRESS 7 TO EXIT
Choice:7
➜ ADS git:(main) ✗