جامعة المنارة
كلية الهندسة
قسم الهندسة المعلوماتية

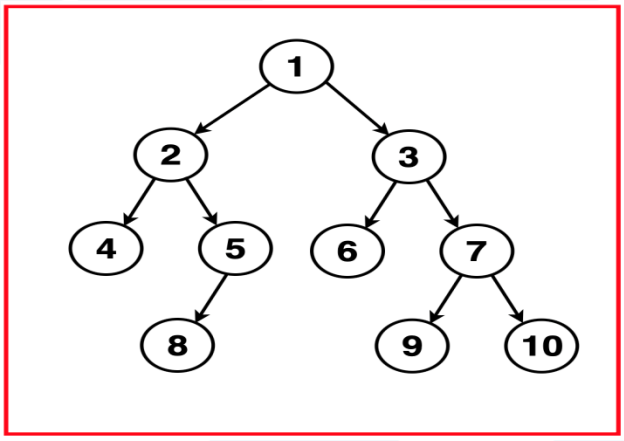
مقرر الخوارزميات و بنى المعطيات1

جلسة العملي السادسة

جَـامعة
المَـنارة
MANARA UNIVERSITY

الأشجار الثنائية

Binary tree traversal

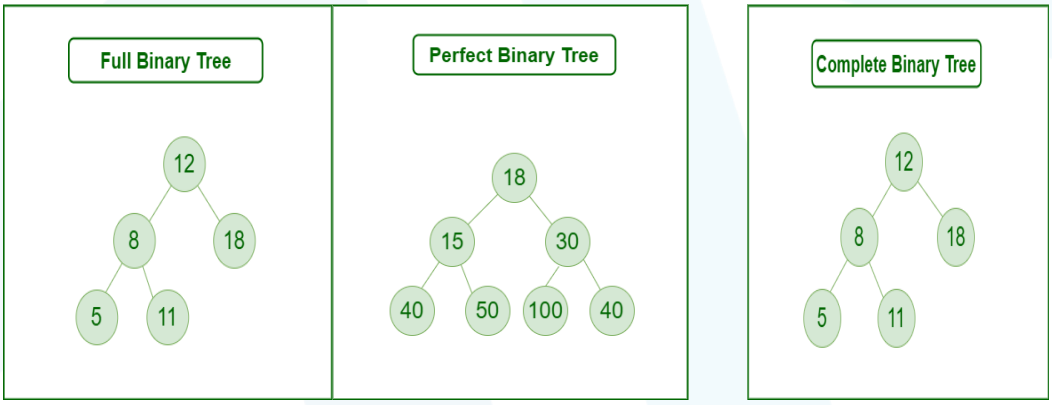


In-order traversal: 4 2 8 5 1 6 3 9 7 10

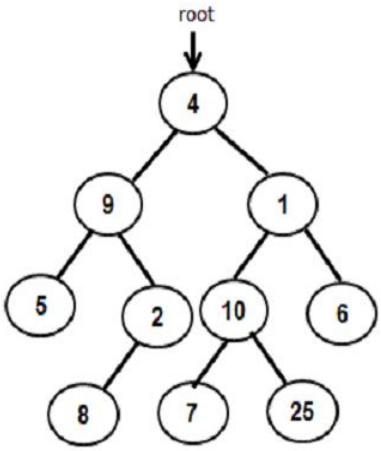
Pre-order traversal : 1 2 4 5 8 3 6 7 9 10

Post-order traversal : 4 8 5 2 6 9 10 7 3 1

Binary Tree Types



أوجد خرج المقطع البرمجي المطبق على الشجرة التالية :

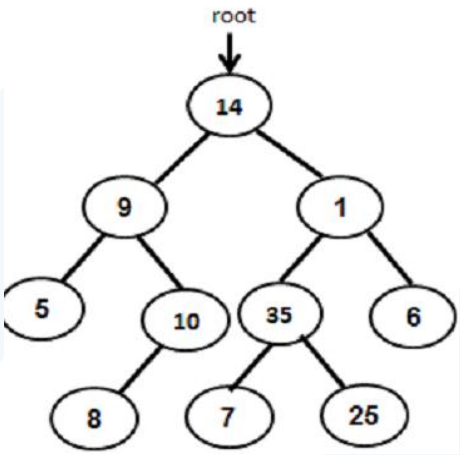



```
Node * p=root;
while(p!=NULL)
{
    cout<< p->data;
    if(p->data%2==0)
        p=p->right;
    else
        p=p->left;
}
```

الخرج :

4 1 10 25

أوجد خرج المقطع البرمجي المطبق على الشجرة التالية :

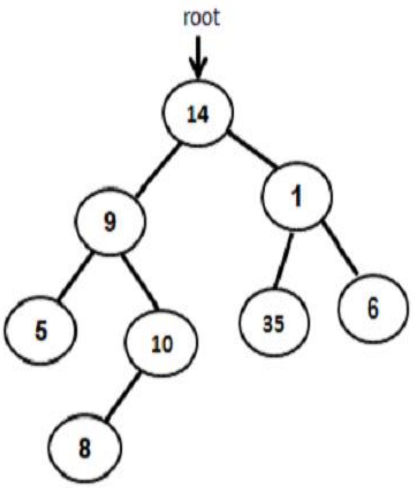



```
Node * ptr=root;
int s=0;
while(ptr!=NULL)
{
    s=s + ptr->data;
    if(ptr->data>10)
        ptr=ptr->left;
    else
        ptr=ptr->right;
}
cout<<s;
```

الخرج

33

أوجد خرج المقطع البرمجي المطبق على الشجرة التالية :



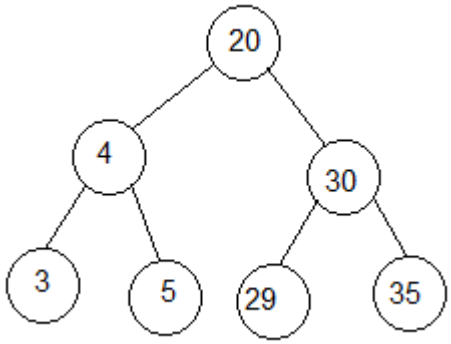

```
Node * p=root->left;
while(p->right!=NULL)
{
    cout<< p->data;
    p=p->right;
}
```

الخرج

9

اكتب برنامجا بلغة C++ يقوم بما يلي:

- بناء شجرة البحث الثنائية الموضحة في الشكل أدناه
- التجول في الشجرة بالأساليب الثلاثة : preorder inorder postorder



الحل:

```
#include <iostream>
using namespace std;
```

تعريف الصف الممثل لعقد الشجرة:

```
class BSTnode{
 public:
 int data;
 BSTnode *left, *right;

 BSTnode(){
  left=right=NULL;
 }
```

```
 BSTnode (int dat, BSTnode *lft=NULL, BSTnode *rgt=NULL){
  data = dat;
  left = lft;
  right = rgt;
 }
};
```

تعريف الصف الممثل للشجرة الثنائية:

```
class BST{
 public:
  BSTnode *root;

  BST(){
   root = NULL;
  }
```

تابع اختبار الشجرة في حال كانت فارغة:

```
  bool isEmpty() { return root==NULL; }
```

تابع إضافة عقدة للشجرة:

```
  void insert (int el, BSTnode* &myroot){
   if (myroot==NULL){
    myroot = new BSTnode (el,0,0);
    cout << el << " inserted\n";
   } else if (el<myroot->data){
    insert (el,myroot->left);
   } else if (el>myroot->data){
    insert (el,myroot->right);
   }
```

}

تابع اختبار انتماء عقدة للشجرة:

```
bool search(int el, BSTnode* &start){
 if (start!=NULL){
  if (el<start->data){
   return search(el,start->left);
  } else if (el>start->data){
   return search (el,start->right);
  } else {
   return true;
  }
 }
 return false;
}
```

تابع مسح عقد الشجرة بشكل inorder:

```
void inorder(node *tree)
{
    if(tree!=NULL)
    {
        inorder(tree->left);
        cout<<tree->data<<"    ";
        inorder(tree->right);
    }
}
```

تابع مسح عقد الشجرة بشكل preorder:

```
void preorder(node *tree)
{
```

```
    if(tree!=NULL)
    {
        cout<<tree->data<<"   ";
        preorder(tree->left);
        preorder(tree->right);
    }
}
```

تابع مسح عقد الشجرة بشكل postorder:

```
void postorder(node *tree)
{
    if(tree!=NULL)
    {
        postorder(tree->left);
        postorder(tree->right);
        cout<<tree->data<<"   ";
    }
}
```

التابع الرئيسي:

```
int main(){
 BST b;
 cout<<"isEmpty "<<b.isEmpty()<<endl;
 b.insert (20,b.root);
 cout<<"isEmpty "<<b.isEmpty()<<endl;
 b.insert (4,b.root);
 b.insert (30,b.root);
 b.insert (3,b.root);
 b.insert (5,b.root);
 b.insert (29,b.root);
```

```
b.insert (35,b.root);

cout << "inorder traversal - ";
b.inorder (b.root);
cout << endl;

cout << "preorder traversal - ";
b.preorder (b.root);
cout << endl;

cout << "postorder traversal - ";
b.postorder (b.root);
cout << endl;



 system ("pause");

return 0;
}
```