جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الأمن السيبراني

DEPARTMENT OF CYBER SECURITY

SUBJECT:

SEARCHING AND SORTING ALGORITHMS

CLASS:

SECOND

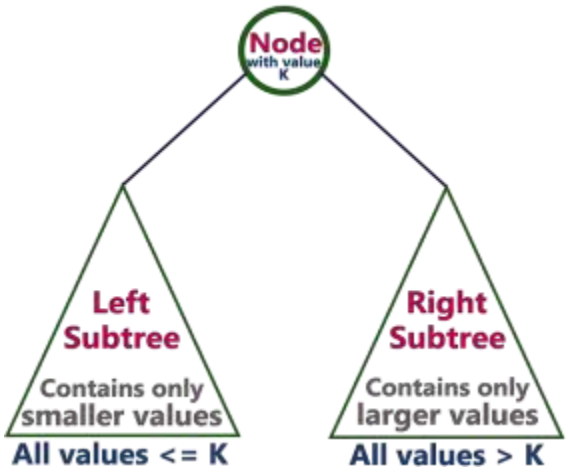
LECTURER: M.SC.MUNTATHER AL-MUSSAWEE

LECTURE: (6 - 7)

BINARY SEARCH TREE

3.1 Binary Search Tree

Binary Search Tree is a binary tree in which every node contains only smaller values in its left subtree and only larger values in its right subtree.

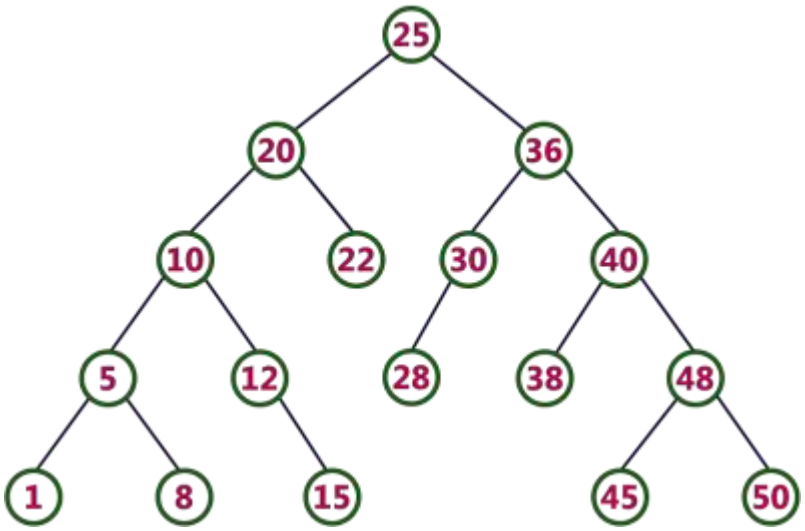


Note: Every Binary Search Tree is a binary tree but **NOT** all the Binary Trees are binary search trees.

Example:

The following tree is a Binary Search Tree. In this tree, left subtree of every node contains nodes with smaller values and right subtree of every node contains

larger values

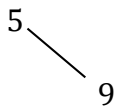


3.2 Insertion to BST

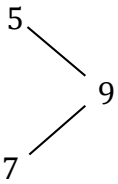
Example: Draw the BST for the following elements:

5,9, 7, 3,8,12, 6, 20

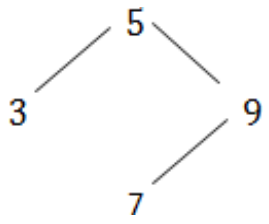
1- Take (5) as a root.
2- Take (9) as a right child because it is greater than the root.



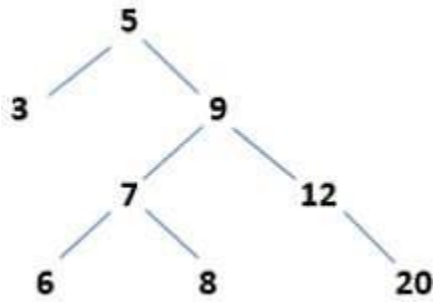
3- The next element (7) is greater than the root so we choose the right branch since it less than 9 so it will be the left child of 9.



4- Take 3 which it less than 5 put it in the left side.



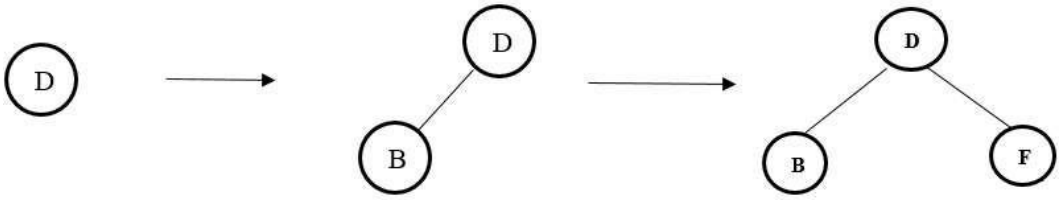
Continue in the same way take the new element and compare it with the tree started from the root , then we will get the final tree as below :

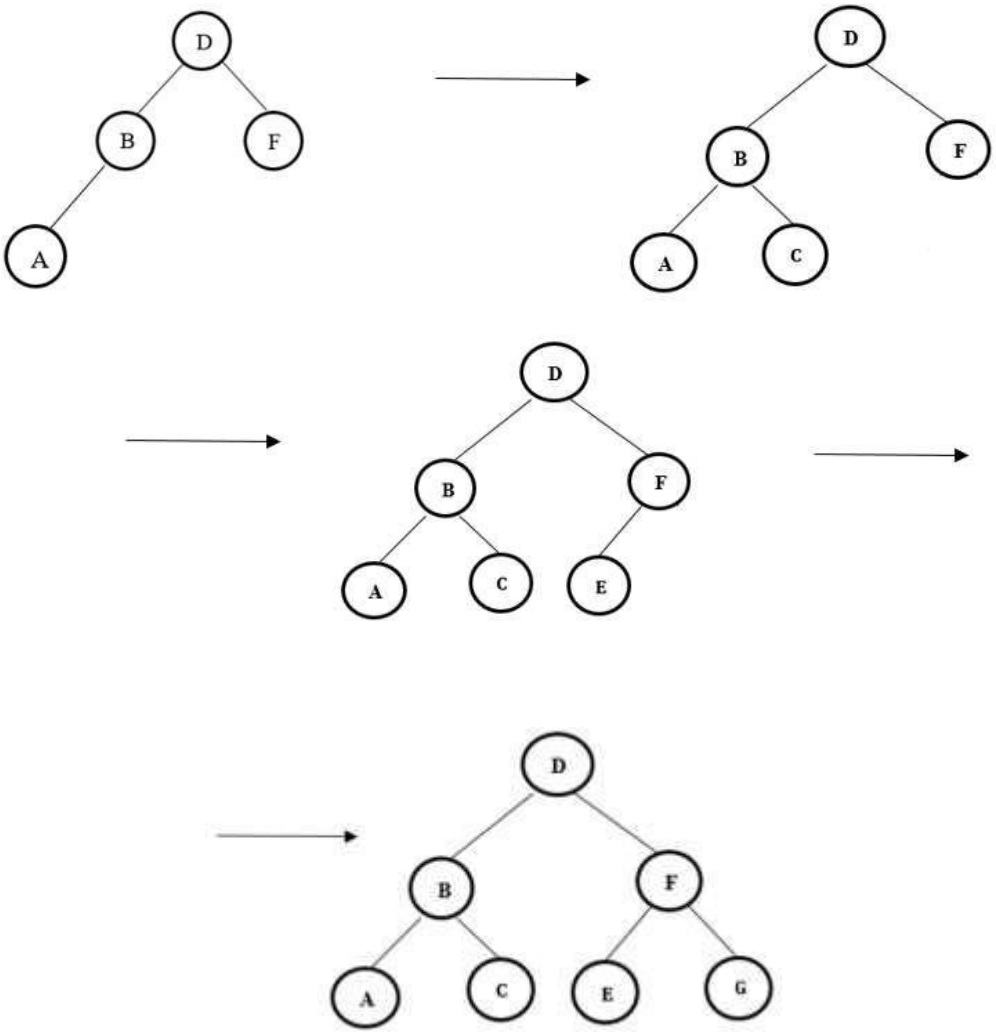


Example: draw the BST for those elements.

NOTE: the ascii code for A=65.

D, B, F, A, C, E, G





Home work: Draw the BST for those elements.

1. B, A, D, C, G, F, E
2. A, B, C, D, E, F, G

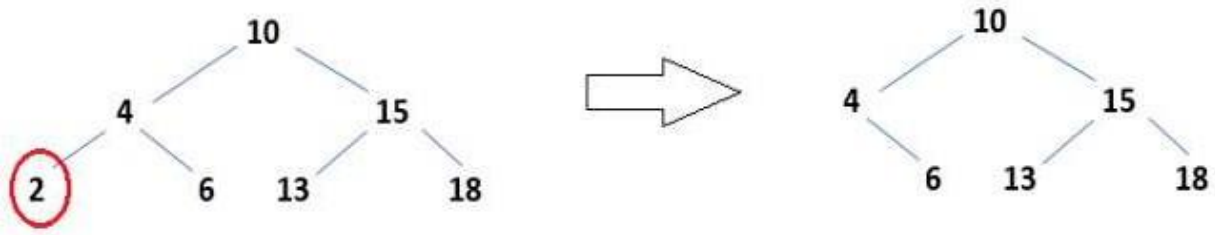
3.3 Deletion Operation in BST

Deleting a node from Binary search tree has following three cases...

Case 1: Deleting a Leaf node (A node with no children)
Case 2: Deleting a node with one child
Case 3: Deleting a node with two children

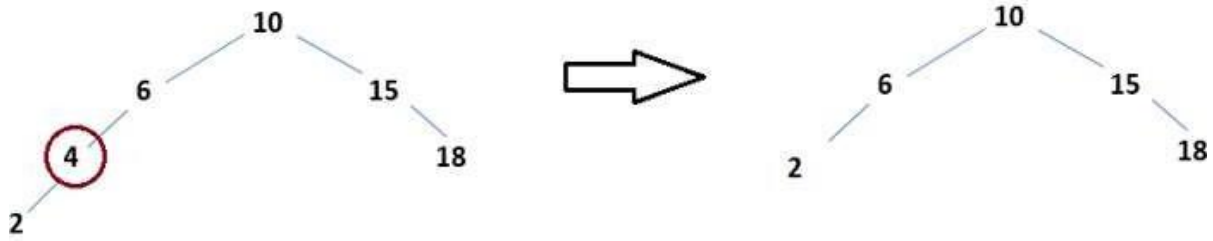
Case 1: Deleting a leaf node

Step 1: Find the node to be deleted.
Step 2: Delete the node and make the father node point to null.



Case 2: Deleting a node with one child:

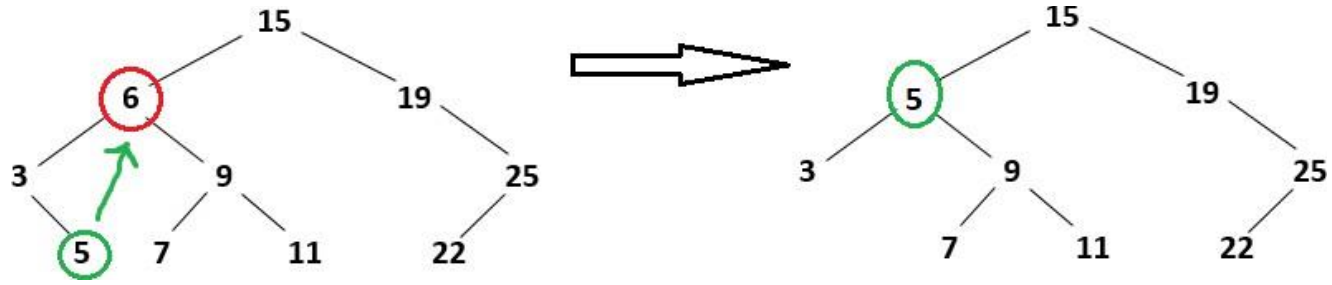
Step 1: Find the node to be deleted.
Step 2: Create a link between its parent and child node.



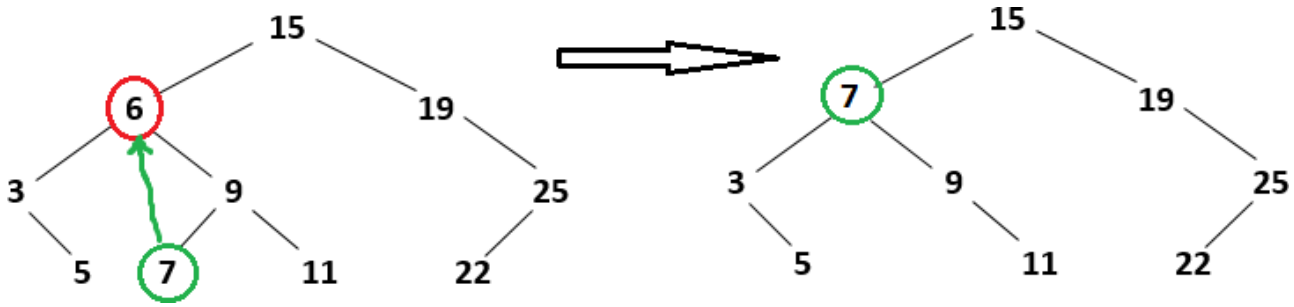
Case 3: Deleting a node with two children

Step 1: Find the node to be deleted.
Step 2: find the **max** node in its left subtree, **OR** the **min** node in its right subtree.

Example: Delete node 6



OR



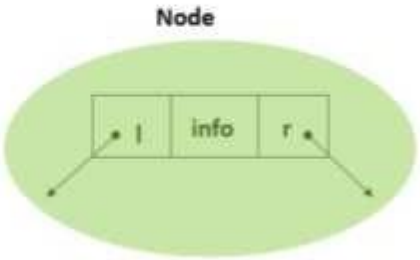
Exercise:

Delete the node 15.

Declaration of Tree by using linked list

نفس طريقة تعريف القائمة الموصولة الفرق اضافة حقل جديد
ليصبح حقل للفرع الايمن وحقل للفرع الايسرمن نوع مؤشر.

```
struct node
{
        int info;
        struct node *l,*r;
};
typedef struct node *nodeptr;
```



Creat the BST

```
void creat(nodeptr& t)
{
  char ch;
  if(t==0)
     {
       t=new node();
       cin>>t->info;
       t->l=0;
       t->r=0;
     }
  cout<<"Do you want to add from left ("<<t->info<<") (Y,N):";
  cin>>ch;
  if(ch=='y')
    creat(t->l);
  cout<<"Do you want to add from right ("<<t->info<<") (Y,N):";
  cin>>ch;
  if(ch=='y')
    creat(t->r);
}
```

عملية خلق اول عقدة
في الشجرة (الجذر)

عملية اضافة عقدة
في الجهة اليسرى

عملية اضافة عقدة
في الجهة اليمنى

Printing the tree by using of the following:

1. In - Order Traversal

```
void inorder(nodeptr t)
{     if(t!=0)
     {
          inorder(t->l);
          cout<<t->info<<' ';
          inorder(t->r);
     }
}
```

2. Pre - Order Traversal

```
void preorder (nodeptr& t)
{     if(t!=0)
     {
          cout<<t->info<<' ';
          preorder(t->l);
          preorder(t->r);
     }
}
```

3. Post - Order Traversal

```
void postorder(nodeptr t)
{   if(t!=0)
    {
          postorder(t->l);
          postorder(t->r);
          cout<<t->info<<' ';
    }
}
```