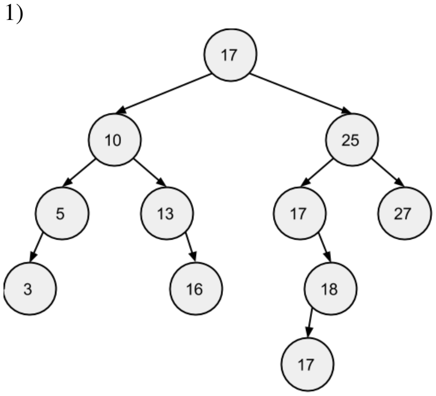
Joanna Klukowska
joannakl@cs.nyu.edu

DNHI Homework 5 Solutions
Trees

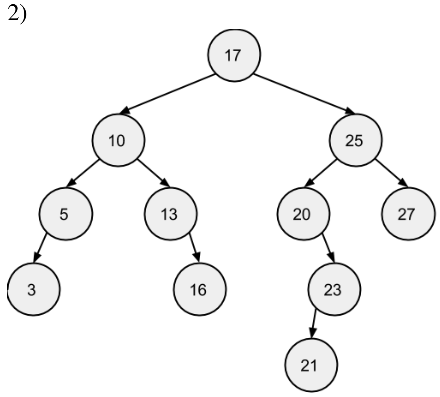
Problem 1

For each of the following trees state what kind of a tree it is (check all that apply).

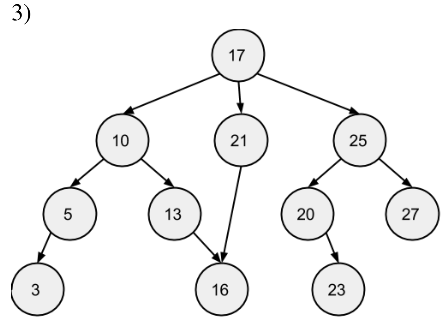
1)



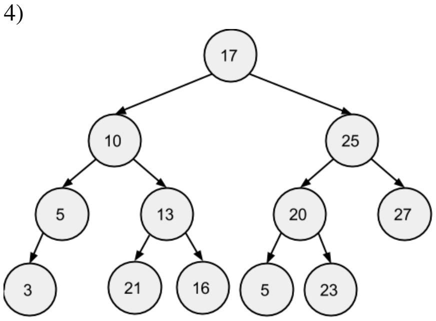
2)



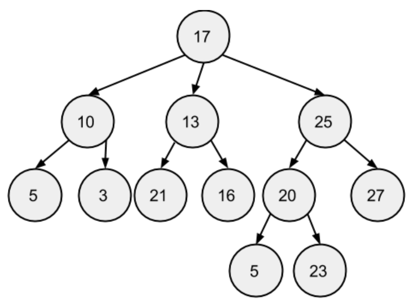
3)



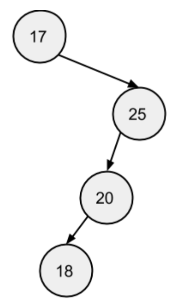
4)



5)



6)



Tree #	Not a tree	General tree	Binary tree	Binary search tree
1		x	x	x
2		x	x	x
3	x			
4		x	x	
5		x		
6		x	x	x

Problem 2

Specify inorder, preorder and postorder traversals of the fourth tree in Problem 1 and the original tree in Problem 3.

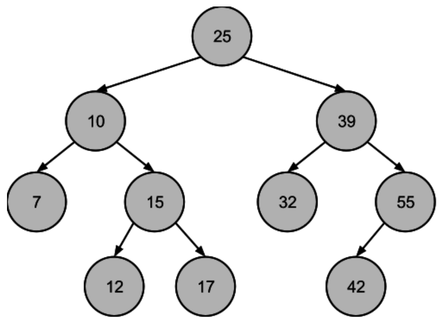
Joanna Klukowska
joannakl@cs.nyu.edu

Tree from problem 1, part 4:
Inorder traversal: 3, 5, 10, 21, 13, 16, 17, 5, 20, 23, 25, 27
Preorder traversal: 17, 10, 5, 3, 13, 21, 16, 25, 20, 5, 23, 27
Postorder traversal: 3, 5, 21, 16, 13, 10, 5, 23, 20, 27, 25, 17

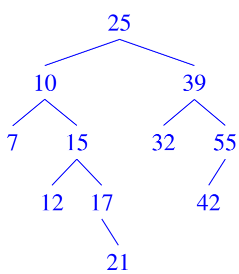
Tree from problem 3:
Inorder traversal: 7, 10, 12, 15, 17, 25, 32, 39, 42, 55
Preorder traversal: 25, 10, 7, 15, 12, 17, 39, 32, 55, 42
Postorder traversal: 7, 12, 17, 15, 10, 32, 42, 55, 39, 25

Problem 3

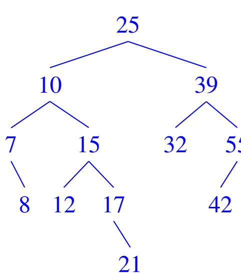
Starting with the binary search tree shown below, show what the tree will look like after each of the following operations. Assume that remove method uses the predecessor when applicable. For each step modify the tree that results from the previous step (NOT the original tree).



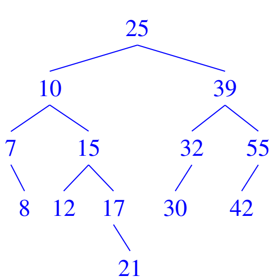
1. `insert(21)`



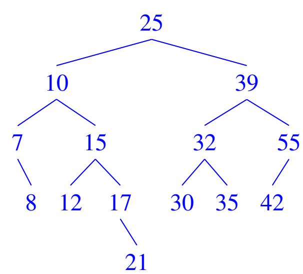
2. `insert(8)`



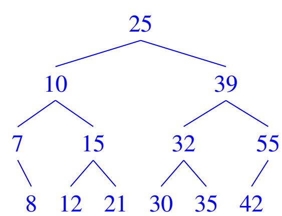
3. `insert(30)`



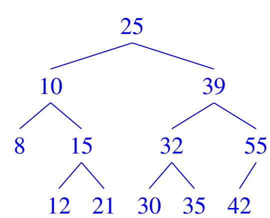
4. `insert(35)`



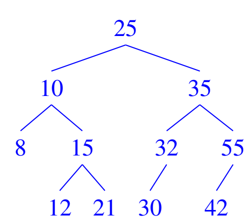
5. `remove(17)`



6. `remove(7)`

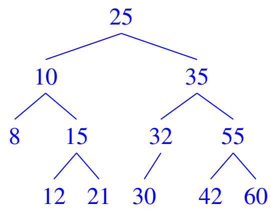


7. `remove(39)`

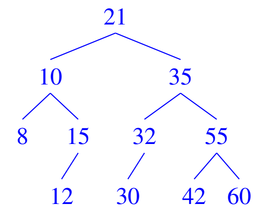


Joanna Klukowska
joannakl@cs.nyu.edu

8. `insert(60)`



9. `remove(25)`



Problem 4

Implement an inorder traversal of a binary tree (this method should work for binary search tree as well) that uses iterative approach. Your method should be a method of a binary tree class. You can assume that there is a private data field called root that points to the root of the tree. You may specify this method using pseudocode, but make sure you are specific. You can assume that on visiting the node you print its content to the standard output.

What changes would you have to make to convert this into a postorder traversal?

Here is the pseudocode for an iterative inorder traversal algorithm.

```
inorderTraversal ( )
  if tree is not empty (root is not null)
    create an empty stack
    set current to root of this tree
    set done to false
    while not done
      if current is not null
        push current onto the stack
        current = current.left
      else if stack is not empty
        current = top of the stack
        remove the item from top of the stack
        process current
        current = current.right
      else
        set done to true
```

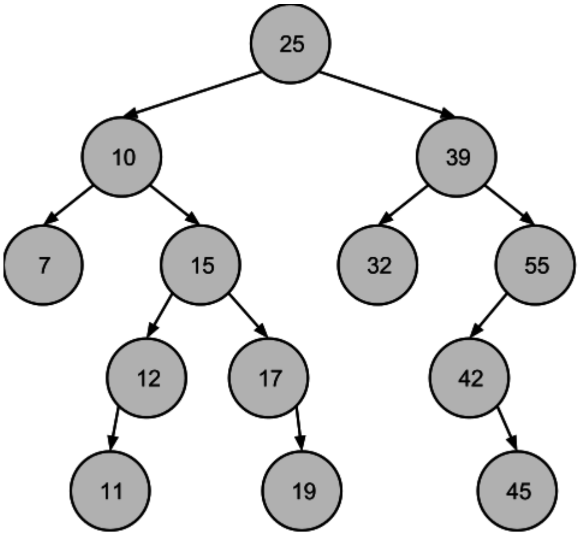
Here is the pseudocode for an iterative postorder traversal algorithm.

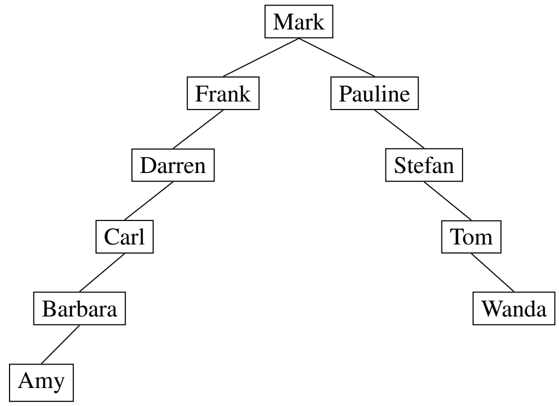
```
postorderTraversal ( )
if tree is not empty (root is not null)
  create an empty stack
  set current to root of this tree
  set done to false
  while not done
    if current is not null
      if current.right is not null
        push current.right onto the stack
      push current onto the stack
      current = current.left
    else if stack is not empty
      current = top of the stack
      remove the item from top of the stack
      if current.right is not null
                    AND stack is not empty
                    AND current.right is equal to the top of the stack
        swap current and top of the stack elements
      else
        process current
    else
      set done to true
```

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 5

Given the following binary search trees, show the structure of the tree after a balancing operation has been performed on it. Assume that when selecting the middle, we always round down (or perform the integer division).



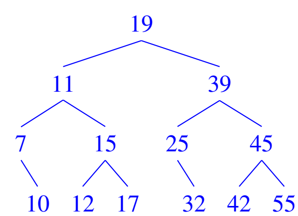


Left tree:

1) perform inorder traversal to obtain a list

```
index:   0  1  2  3  4  5  6  7  8  9 10 11 12
value:   7 10 11 12 15 17 19 25 32 39 42 45 55
```

2) use the recursive algorithm that picks middle of the array element to add to the tree and then, middles of the middles, etc.

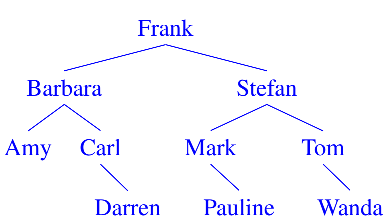


Right tree:

1) perform inorder traversal to obtain a list

```
index:    0     1      2     3      4     5     6       7     8    9
value:   Amy Barbara Carl Darren Frank Mark Pauline Stefan Tom Wanda
```

2) use the recursive algorithm that picks middle of the array element to add to the tree and then, middles of the middles, etc.



Problem 6

Given the trees in Problem 5, show their preorder and postorder traversals.

Joanna Klukowska
joannakl@cs.nyu.edu

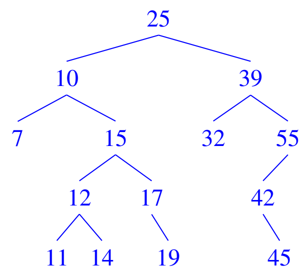
Left tree:
Preorder traversal: 25, 10, 7, 15, 12, 11, 17, 19, 39, 32, 55, 42, 45
Postorder traversal: 7, 11, 12, 19, 17, 15, 10, 32, 45, 42, 55, 39, 25

Right tree:
Preorder traversal: Mark, Frank, Darren, Carl, Barbara, Amy, Pauline, Stefan, Wanda, Tom
Postorder traversal: Any, Barbara, Carl, Darren, Frank, Tom, Wanda, Stefan, Pauline, Mark

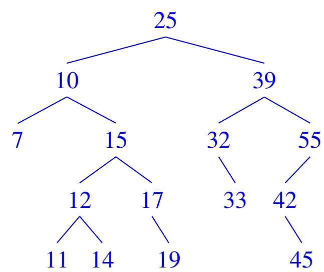
Problem 7

Given the left tree in Problem 5, show the tree after the following operations. Assume that remove operations use the successor to replace a removed node when appropriate.

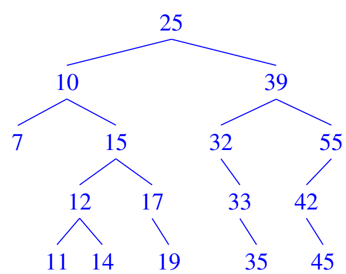
1. `insert(14)`



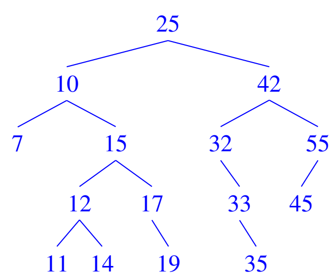
2. `insert(33)`



3. `insert(35)`

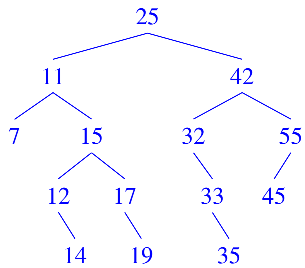


4. `remove(39)`

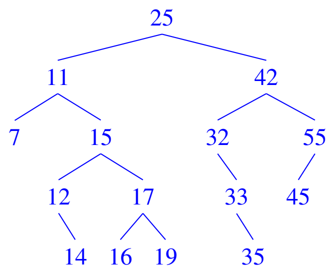


Joanna Klukowska
joannakl@cs.nyu.edu

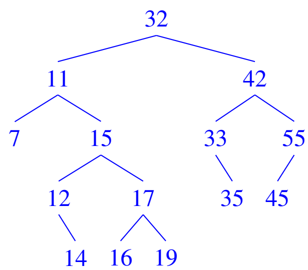
5. `remove(10)`



6. `insert(16)`



7. `remove(25)`



Problem 7

Write a method of a binary tree that determines the size of the tree. You can write pseudocode. You cannot assume that there is a data field storing the size of the tree.

```
int size ()
  return size ( head )

int size ( Node n )
  if n == null
    return 0
  else return 1 + size ( n.left ) + size ( n.right )

```

Problem 8

Write a method of a binary tree that determines the number of leaves in the tree. You can write pseudocode.

```
int countLeaves ()
  return countLeaves ( head )

int countLeaves ( Node n )
  if n == null //called with null, don't count
    return 0
  if n.left == null && n.right == null //we have a leaf
    return 1
  else return countLeaves ( n.left ) + countLeaves ( n.right )

```

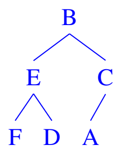
Joanna Klukowska
joannakl@cs.nyu.edu

Problem 9

Draw a binary tree for which the inorder and preorder traversals are as follows:

inorder: F E D B A C

preorder: B E F D C A



Problem 10

Given a node in a binary tree, write a recursive method that computes the height of that node. The nodes do not store any height information. You may use pseudocode. The height of a node is the number of edges from the node to the deepest leaf.

```
int height ( Node n )
  if n == null
    return -1
  else
    return 1 + max( height(n.left), height(n.right) )
```

Problem 11

Given a binary tree with 3 levels (level 0, level 1 and level 2) what is the largest number of nodes that the tree may contain? what is the smallest number of nodes that the tree may contain?

largest: full tree with 7 nodes
smallest: 3 nodes (linked list)

Problem 12

Write a method of a binary (search) tree class that returns the sum of all the numbers stored in the nodes. Write another method that returns the sum of the numbers stored in the leaf-nodes. Write another method that returns the sum of the numbers stored at even numbered levels (assume that the root is at level 0, which is even).

Assume that the nodes store integers.

```
int sum ()
  return sum ( root )

int sum ( Node n )
  if n == null
    return 0
  else
    return n.value + sum( n.left ) + sum ( n.right )
```

```
int sumOfLeaves ()
  if root == null
    return 0
  if root.left == null && root.right == null
    return root.value
  return sumOfLeaves ( root )
```

Joanna Klukowska
joannakl@cs.nyu.edu

```
int sumOfLeaves ( Node n )
  if n == null
    return 0
  if n.left == null && n.right == null
    return n.value
  else
    return sumOfLeaves( n.left ) + sumOfLeaves ( n.right )

```

```
int sumOfEvenLevels ()
  return sumOfEvenLevels ( root, true )

int sumOfEvenLevels ( Node n, boolean addLevel  )
  if n == null
    return 0
  if addLevel
    return n.value + sumOfLeaves( n.left, !addLevel ) + sumOfLeaves ( n.right, !addLevel )
  else
    return sumOfLeaves( n.left, !addLevel ) + sumOfLeaves ( n.right, !addLevel )

```

Problem 13

Write a method of a binary search tree class that converts the tree to its mirror image (i.e., swaps left and right child for each node). Is the resulting tree a binary search tree?

```
void mirror ()
  mirror ( root )

void mirror ( Node n )
  if n == null
    return
  else
    Node tmp = n.left
    n.left = n.right
    n.right = tmp
    mirror ( n.right )
    mirror ( n.left )

```

Problem 14

Given a sorted array (increasing order) of integers, write an algorithm that creates a binary search tree of minimal height.

```

isertNodes( listOfValues, first, last )

  if (first == last)
    insert( listOfValues[first] )

  else if (first+1 == last)
    insert( listOfValues[first] )
    insert( listOfValues[last] )

  else
    mid = (first + last) / 2
    insert ( listOfValues[mid] )
    insertNodes ( first, mid-1 )
    insertNodes ( mid+1, last )
```