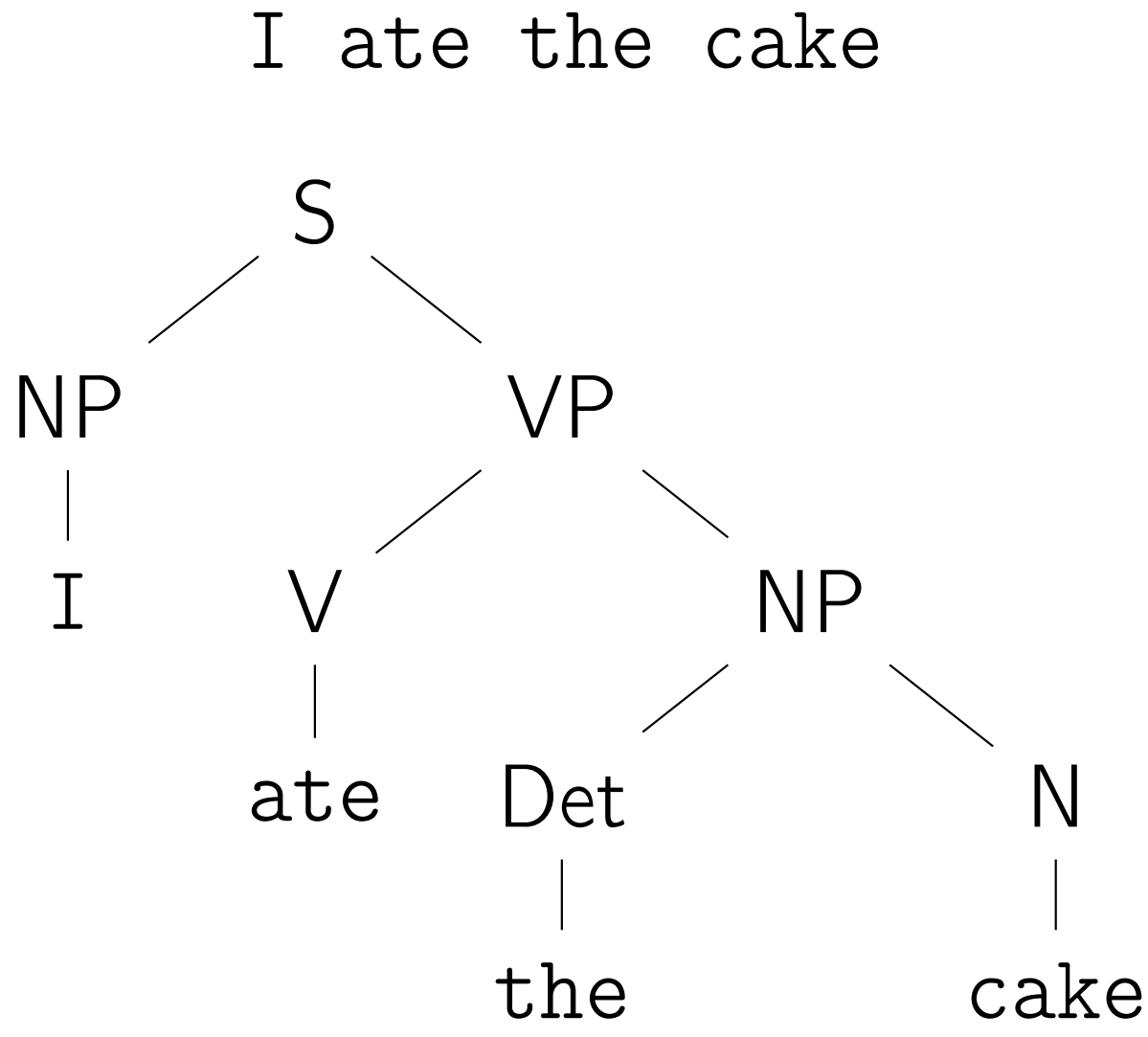
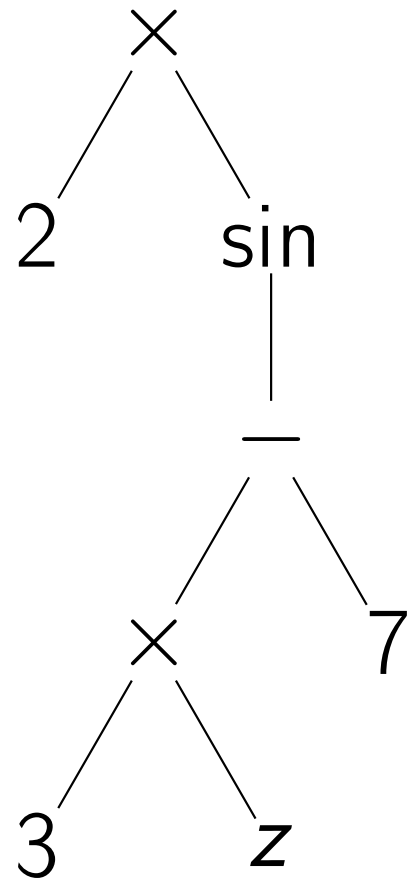
Syntax Tree for a Sentence

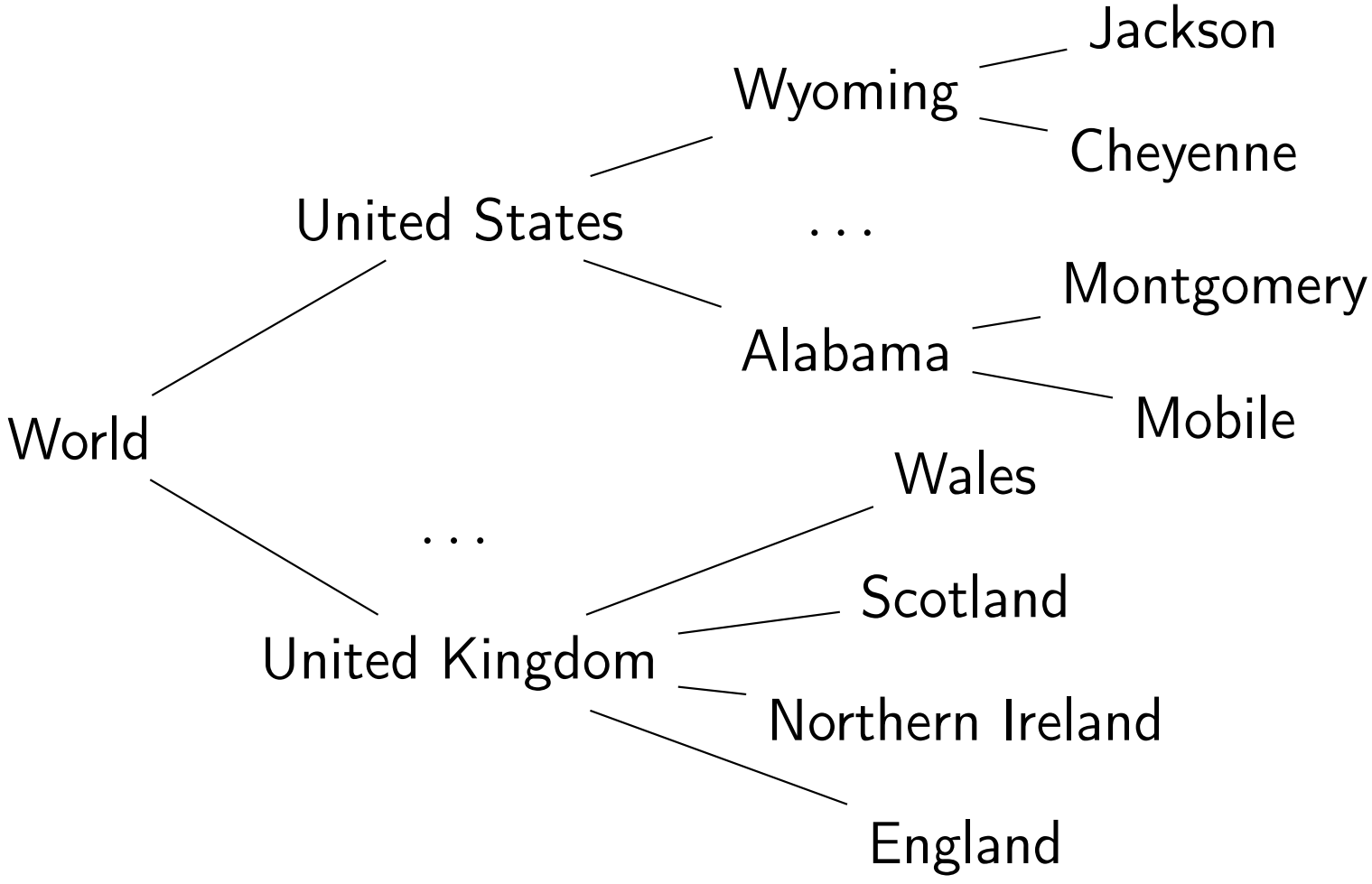


Syntax tree for an Expression

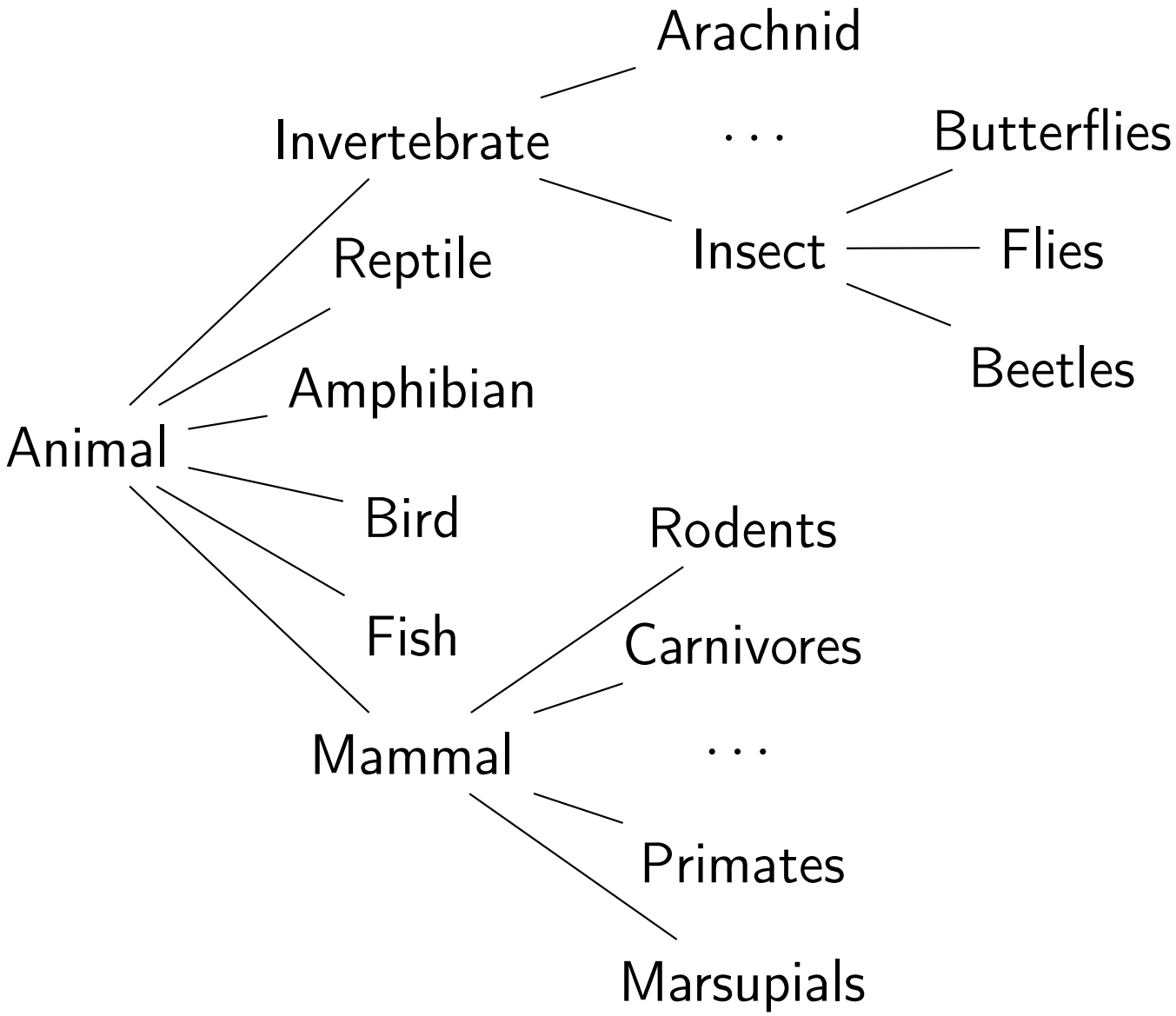
$$2\sin(3z - 7)$$



Geography Hierarchy

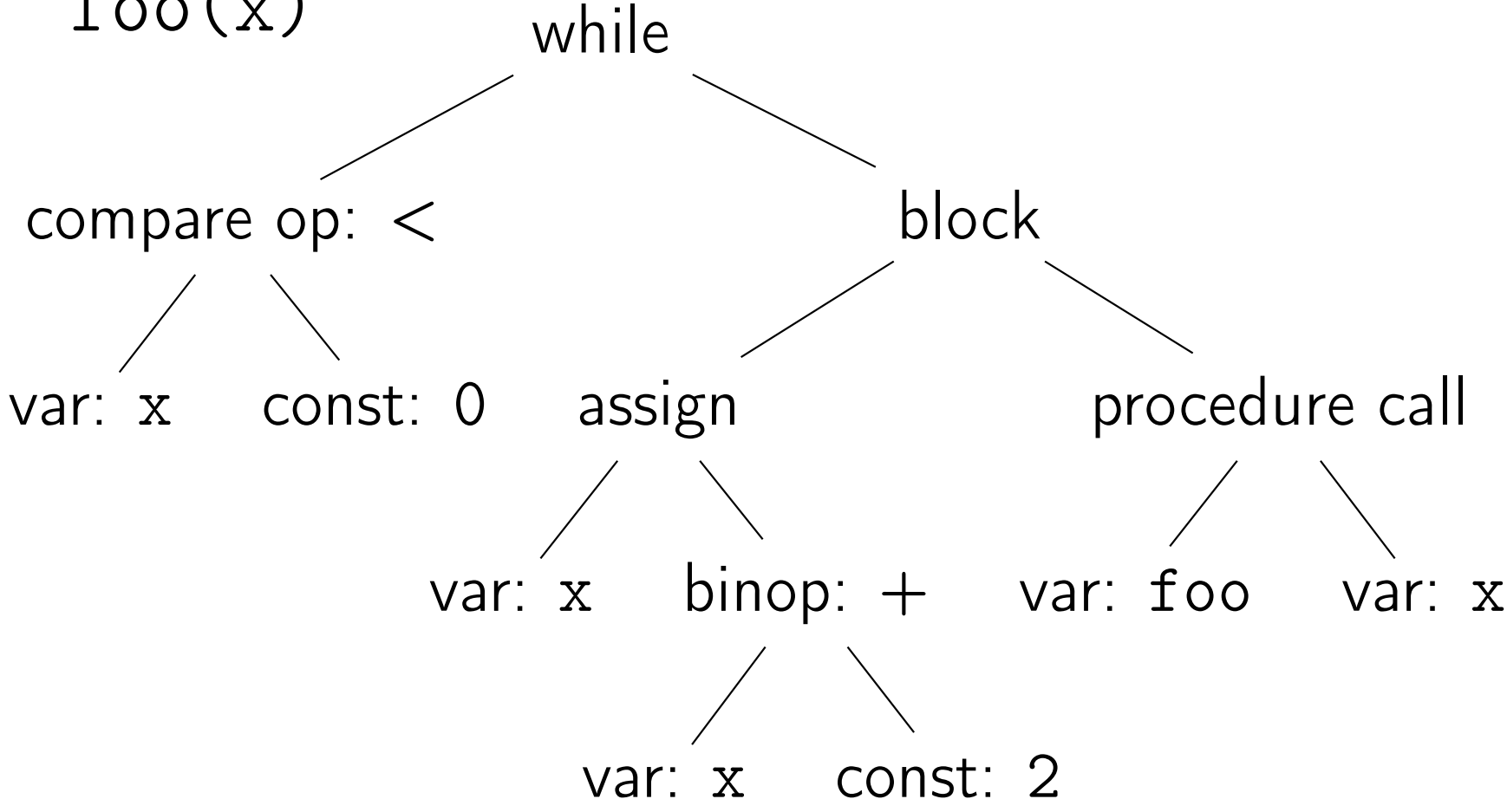


Animal Kingdom (partial)

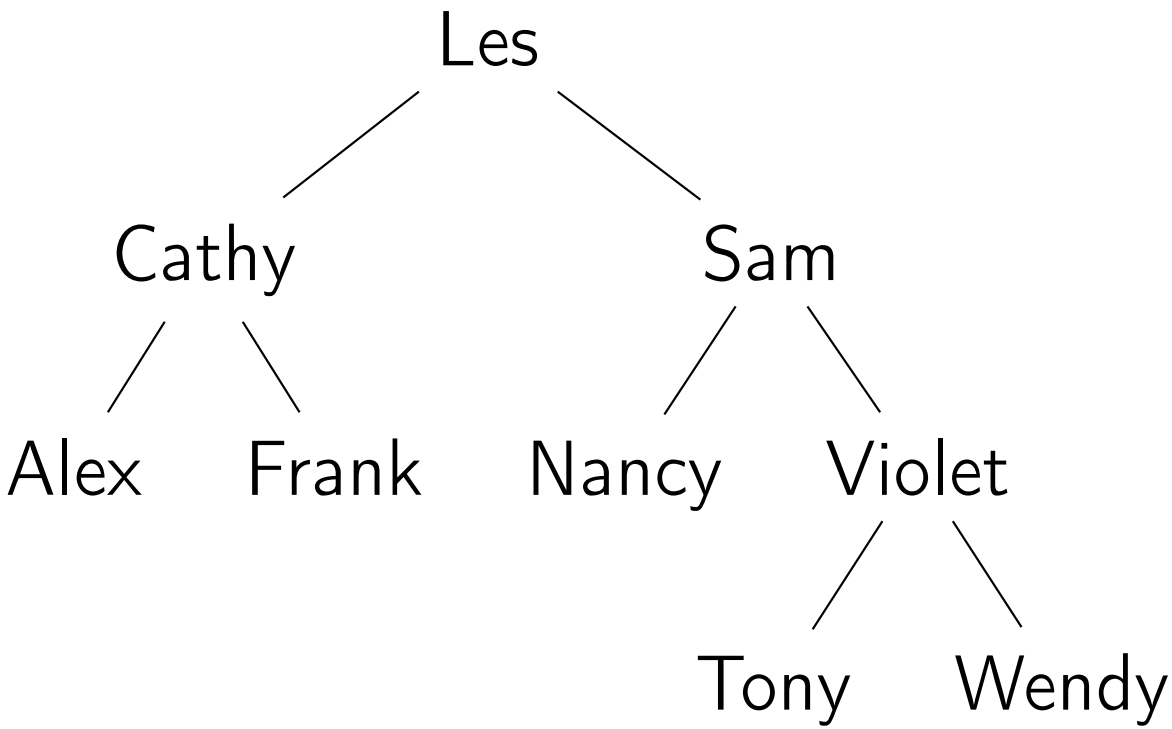


Abstract Syntax Tree for Code

```
while x < 0:
  x = x + 2
  foo(x)
```



Binary Search Tree



Definition

A Tree is:

- empty, or
- a node with:
  - a key, and
  - a list of child trees.

Simple Tree

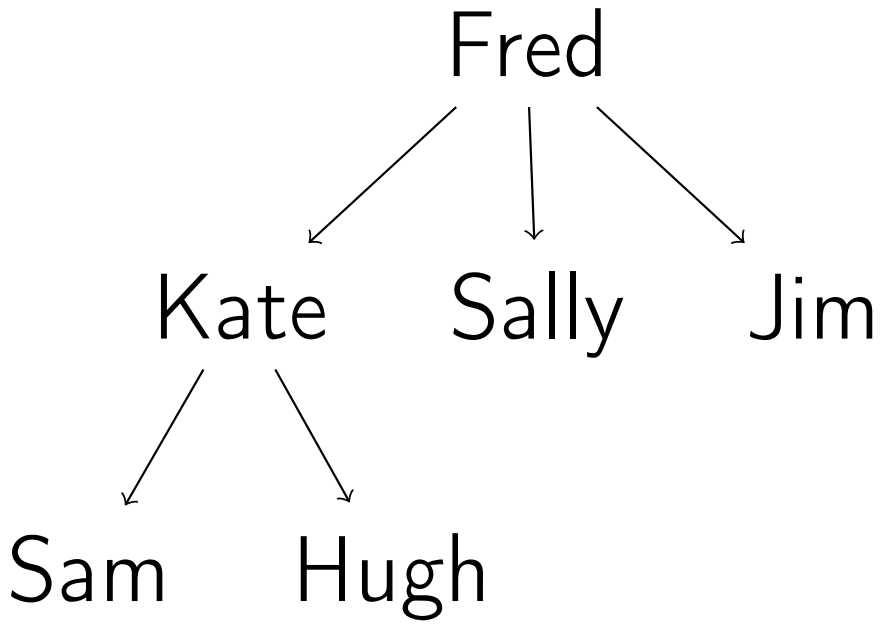
Empty tree:

Tree with one node:

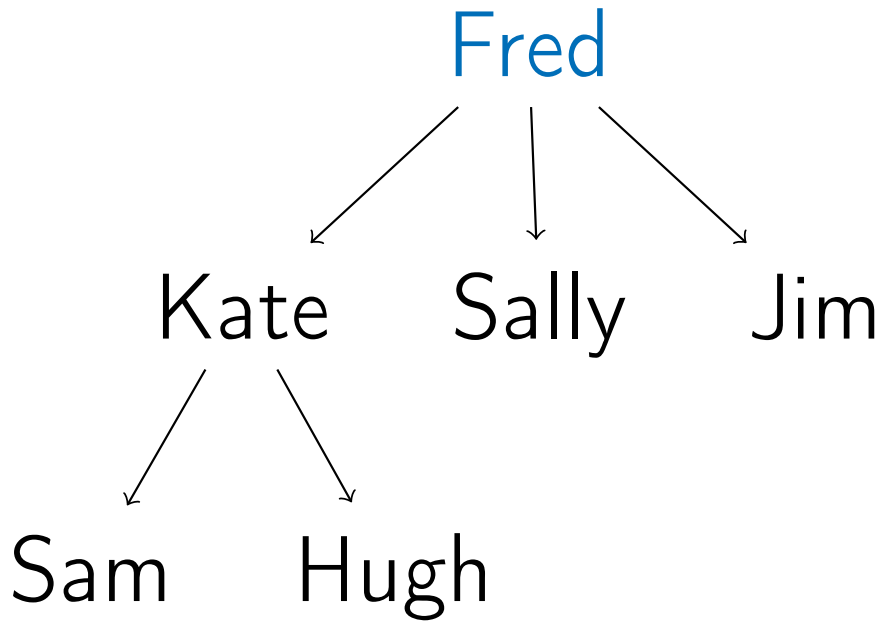
Fred

Tree with two nodes:

Terminology



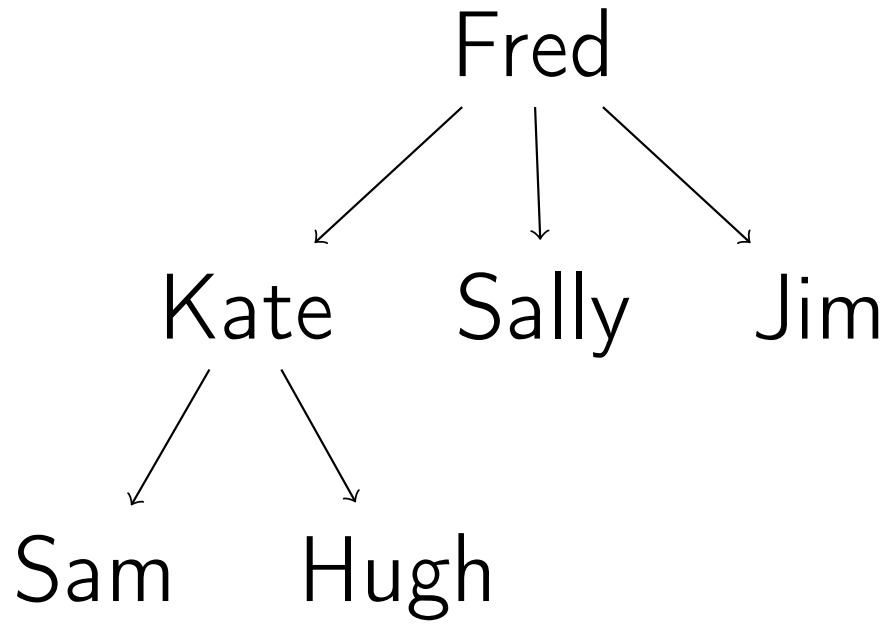
Terminology



***Root*:**

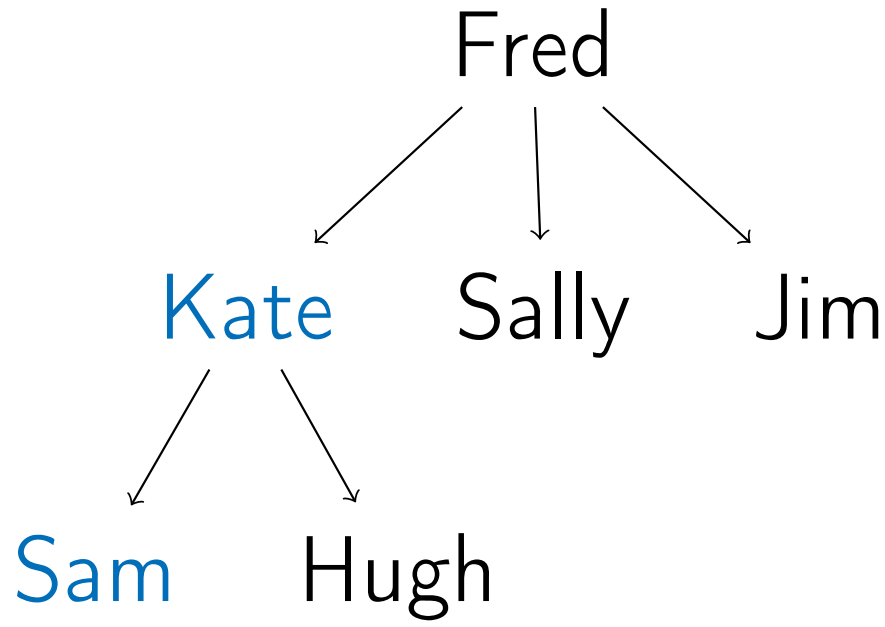
top node in the tree

Terminology



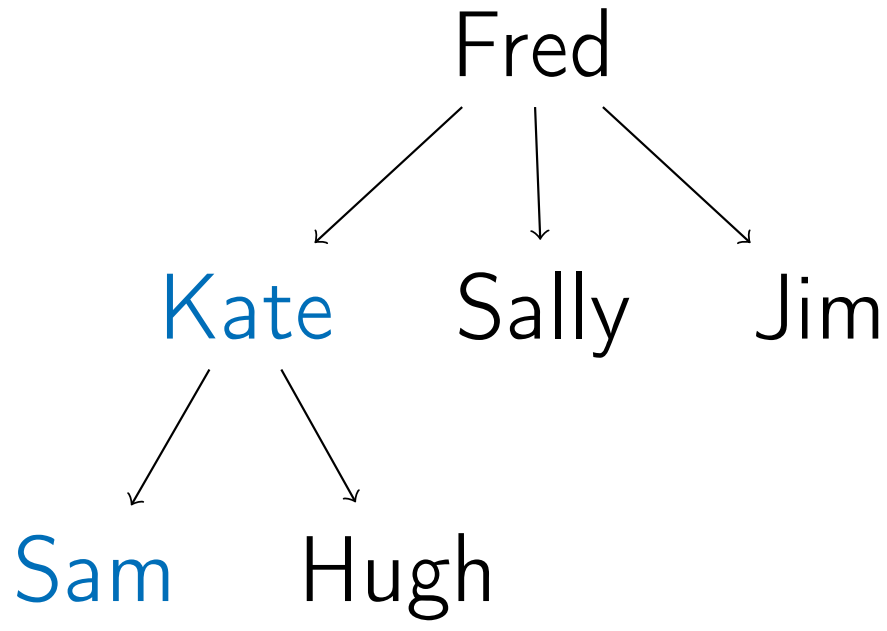
A *child* has a line down directly from a *parent*

Terminology



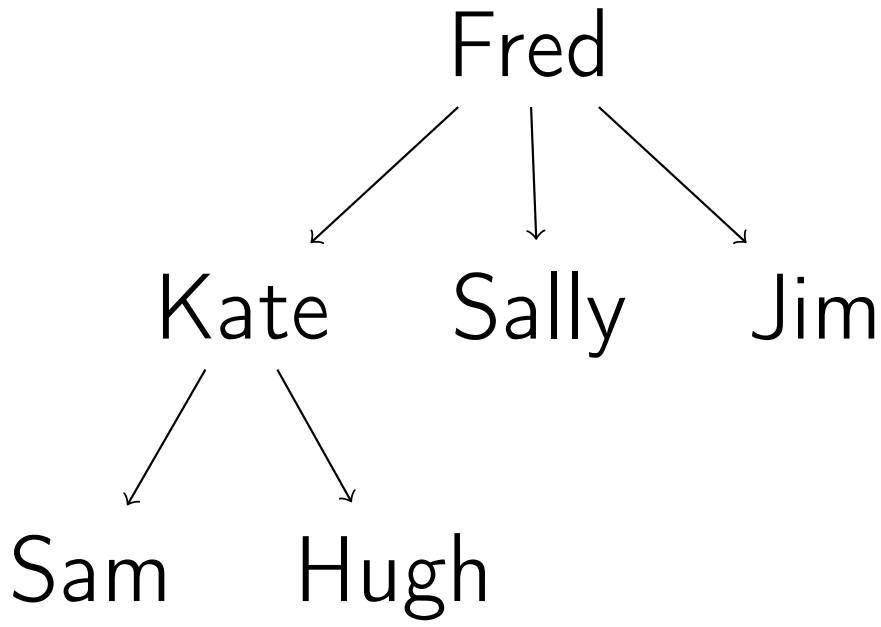
Kate is a *parent* of Sam

Terminology



Sam is a *child* of Kate

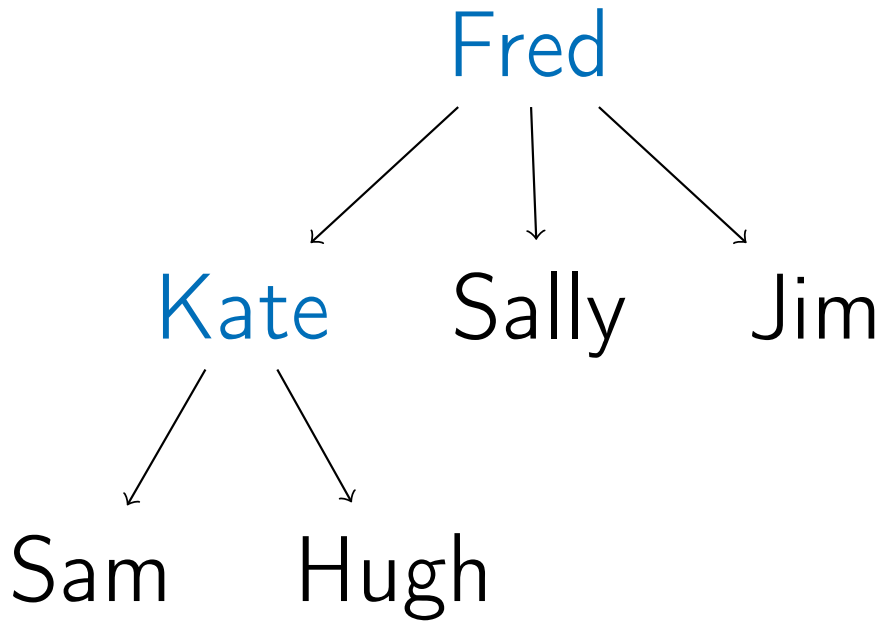
Terminology



Ancestor:

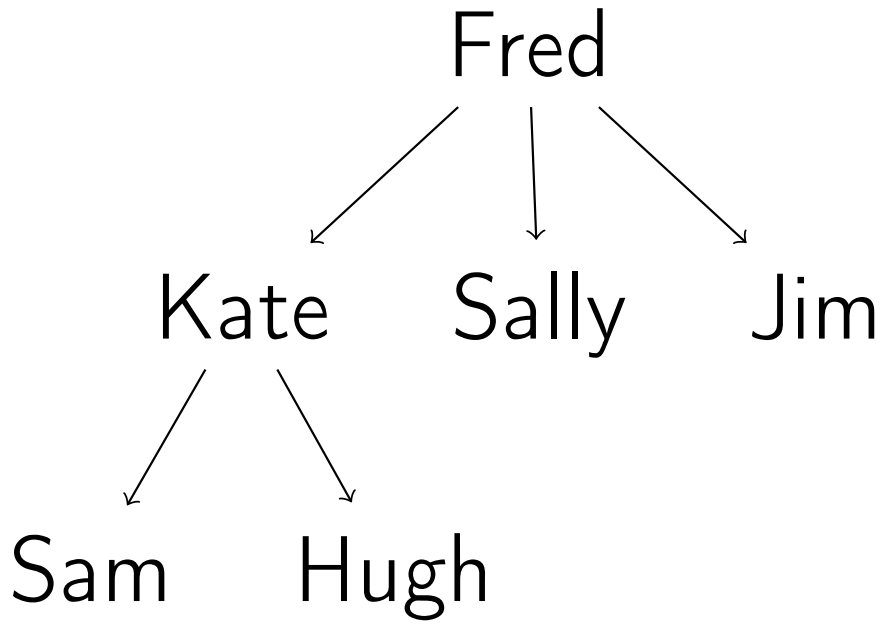
parent, or parent of parent, etc.

Terminology



***Ancestors* of Sam**

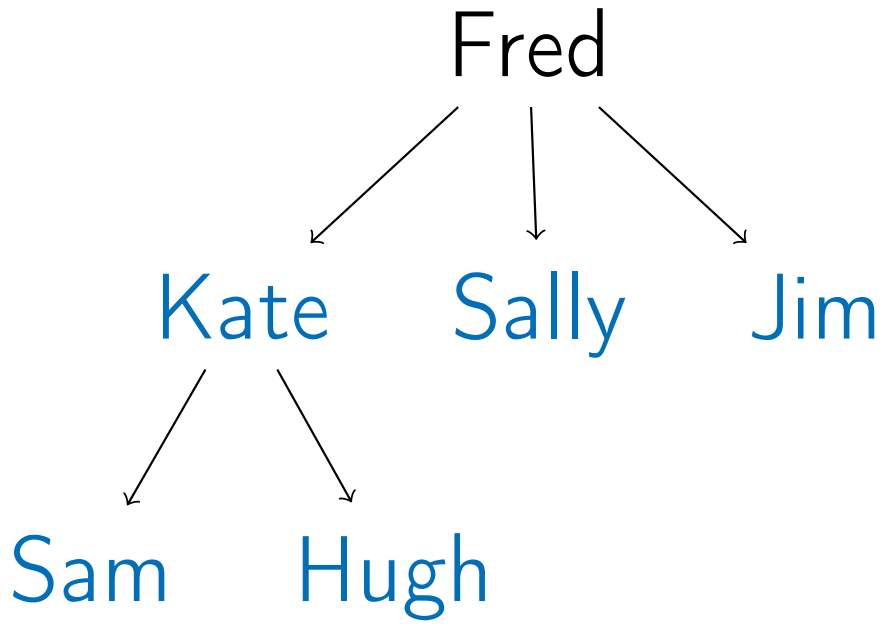
Terminology



Descendant:

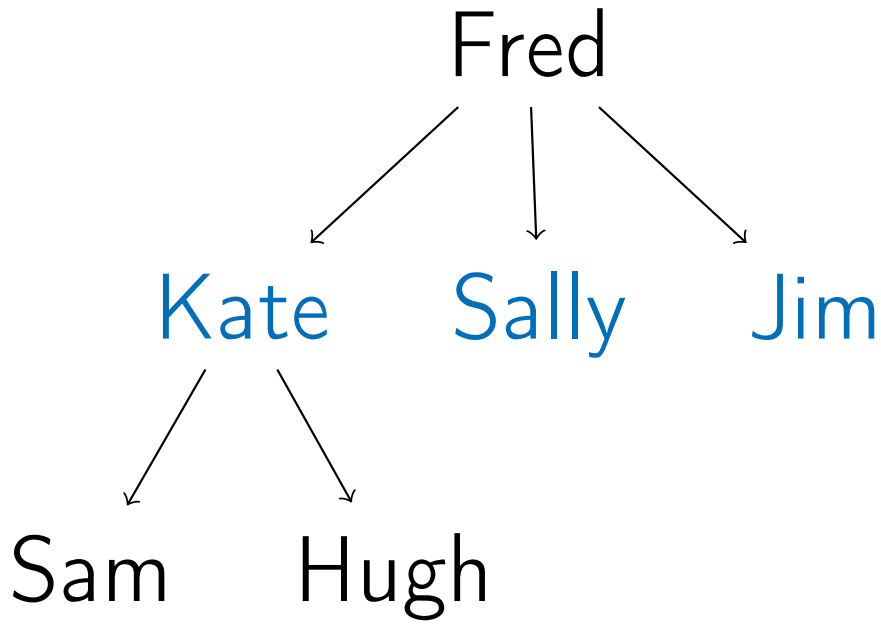
child, or child of child, etc.

Terminology



***Descendants* of Fred**

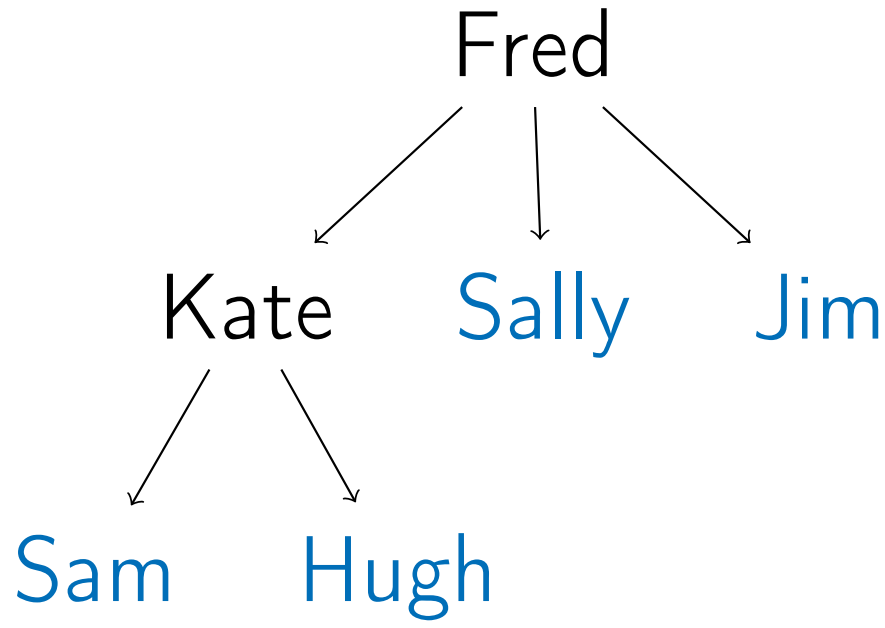
Terminology



***Sibling*:**

sharing the same parent

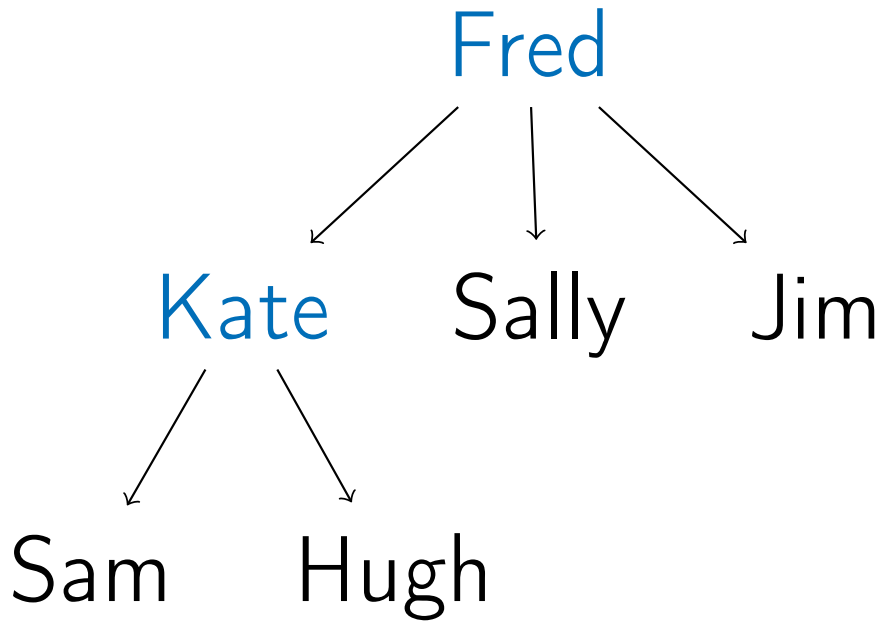
Terminology



***Leaf*:**

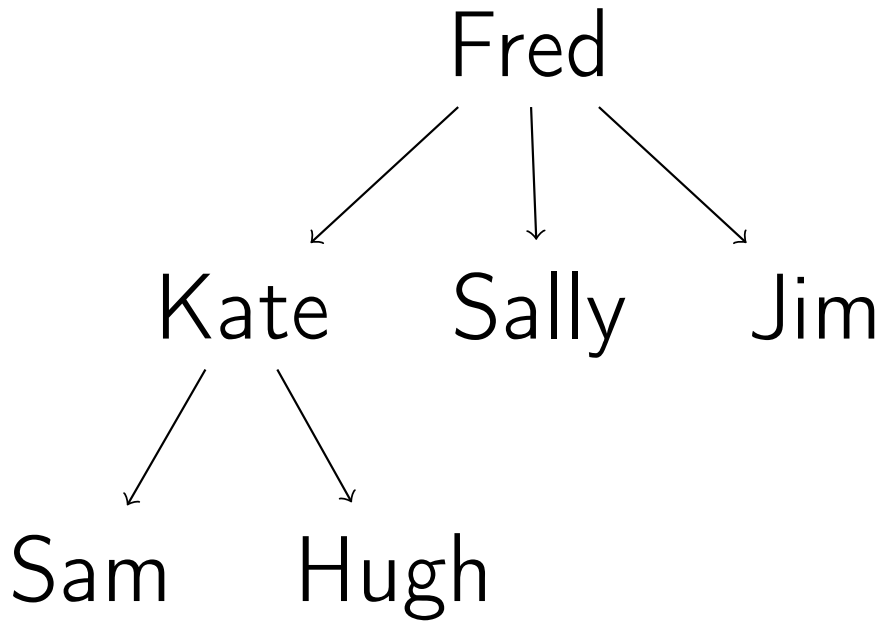
node with no children

Terminology



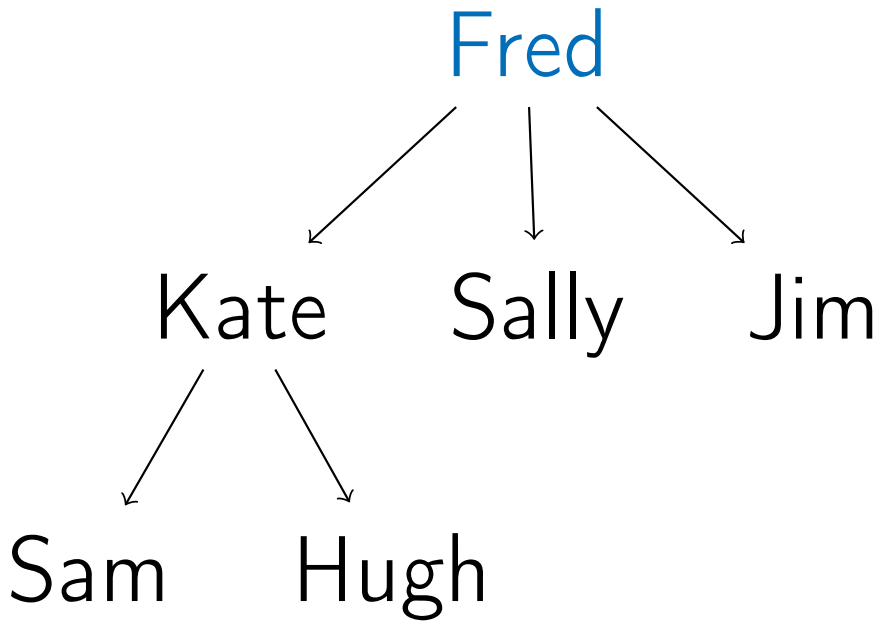
Interior node (non-leaf)

Terminology



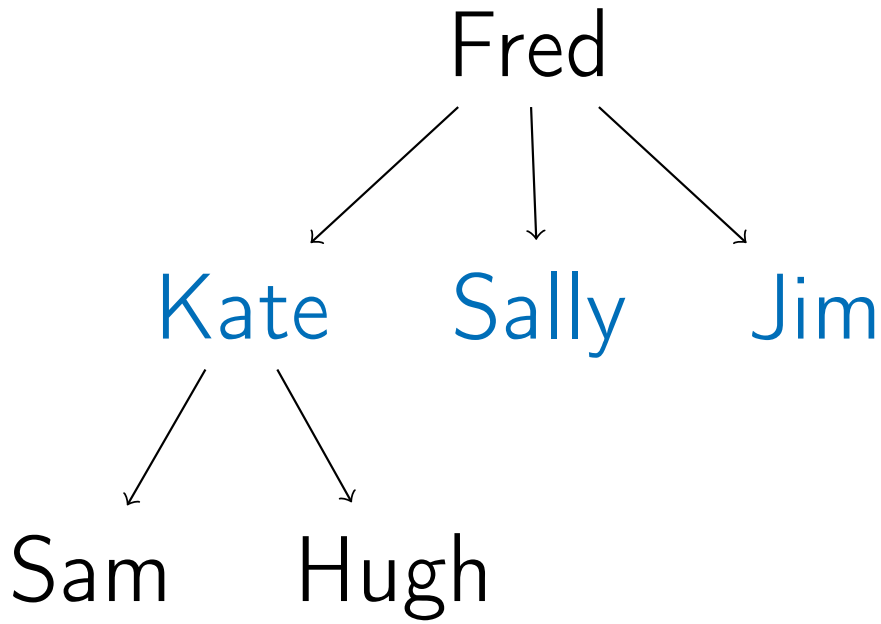
***Level*: 1+ num edges between root and node**

Terminology



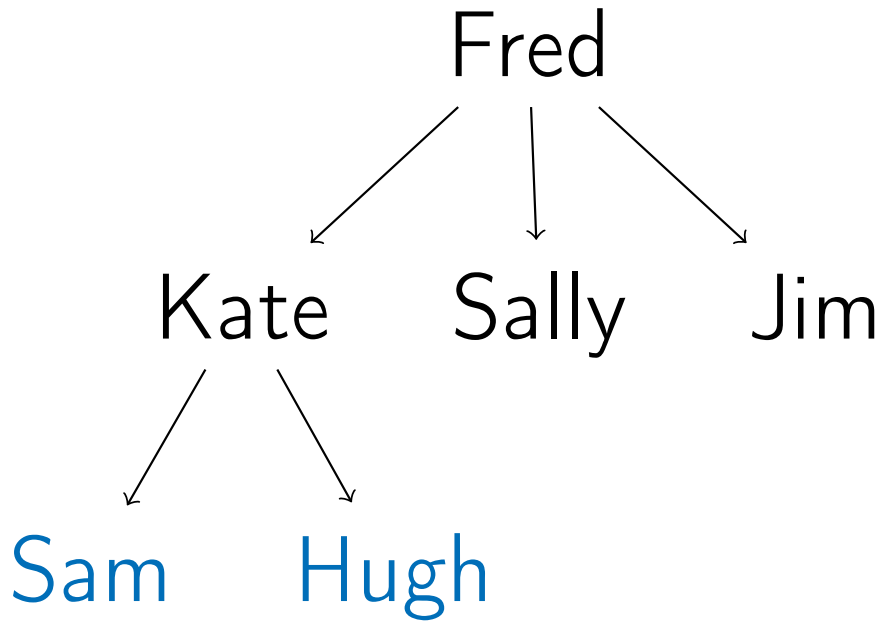
Level 1

Terminology



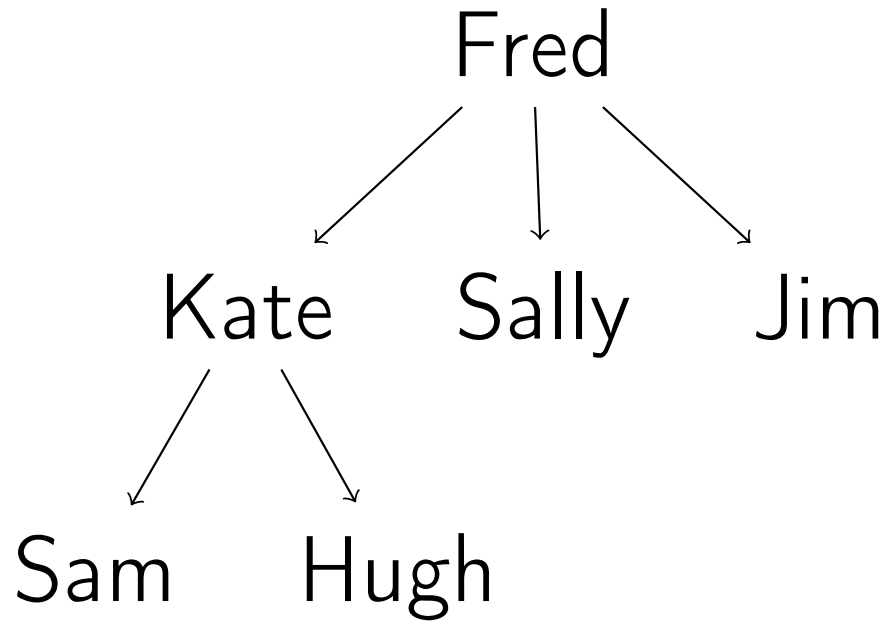
Level 2

Terminology



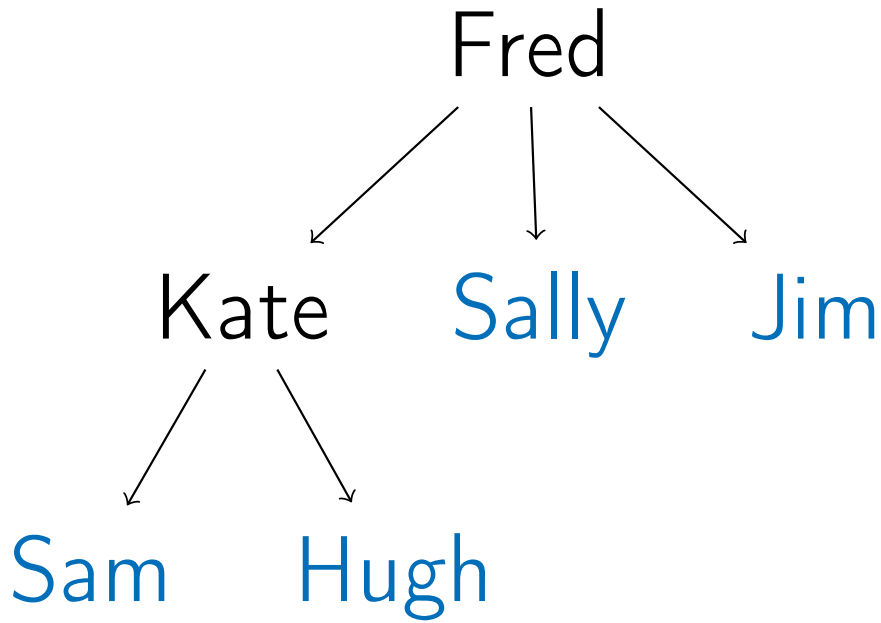
Level 3

Terminology



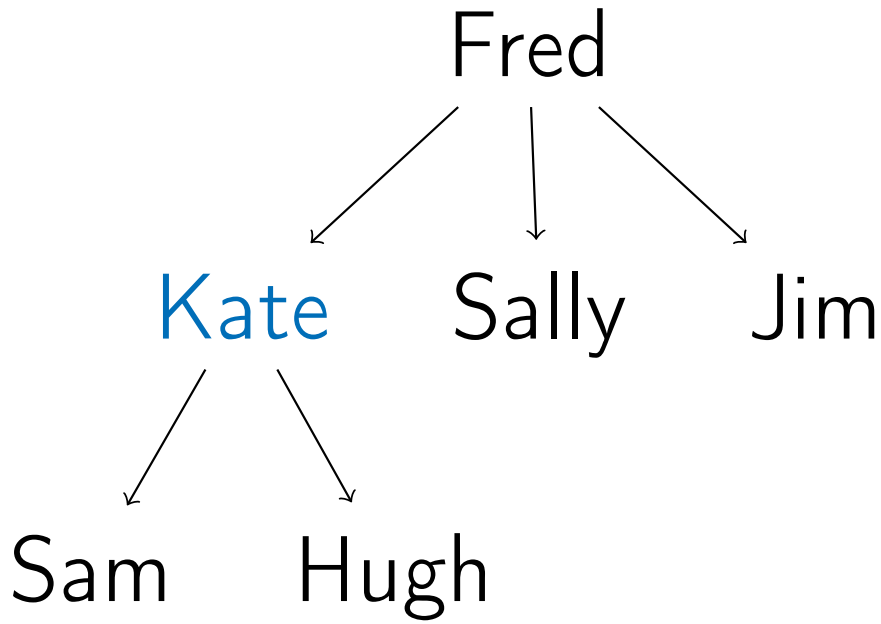
***Height*: maximum depth of subtree node and farthest leaf**

Terminology



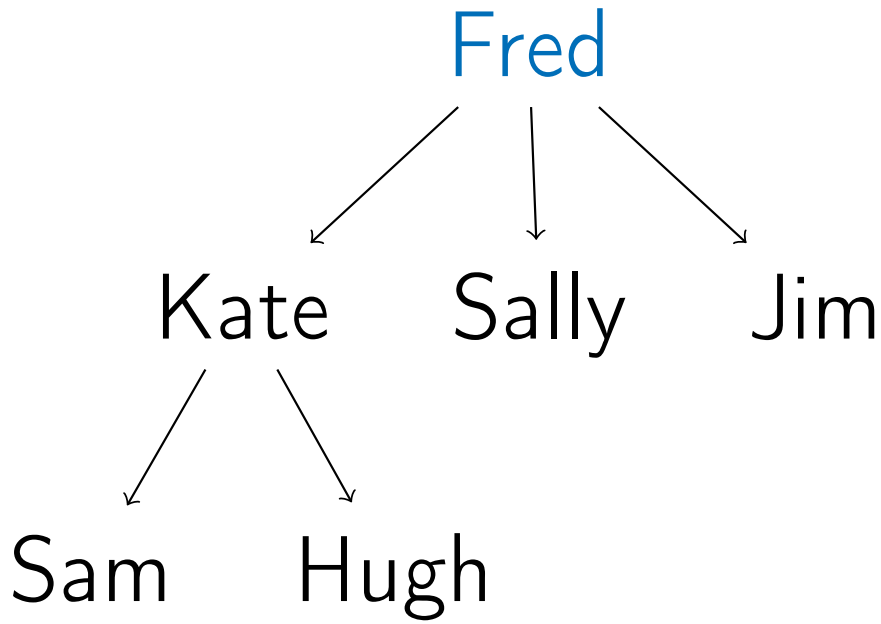
Height 1

Terminology



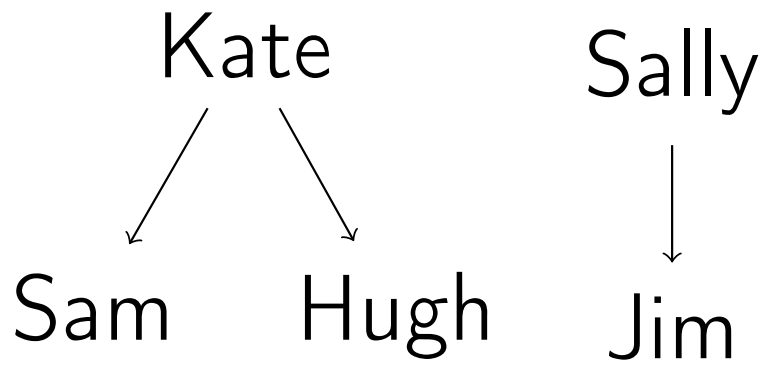
Height 2

Terminology



Height 3

Terminology



***Forest*:**
collection of trees

Node contains:

- `key`
- `children`: list of children nodes
- (optional) `parent`

For binary tree, node contains:

- `key`
- `left`
- `right`
- (optional) `parent`

Height(*tree*)

```
if tree = nil :
   return 0
return 1 + Max(Height(tree.left),
                Height(tree.right))
```

Size(*tree*)

```
if tree = nil
   return 0
return 1 + Size(tree.left) +
           Size(tree.right)
```

Walking a Tree

Often we want to visit the nodes of a tree in a particular order.

Walking a Tree

Often we want to visit the nodes of a tree in a particular order.

For example, print the nodes of the tree.

Walking a Tree

Often we want to visit the nodes of a tree in a particular order.
For example, print the nodes of the tree.

- Depth-first: We completely traverse one sub-tree before exploring a sibling sub-tree.

Walking a Tree

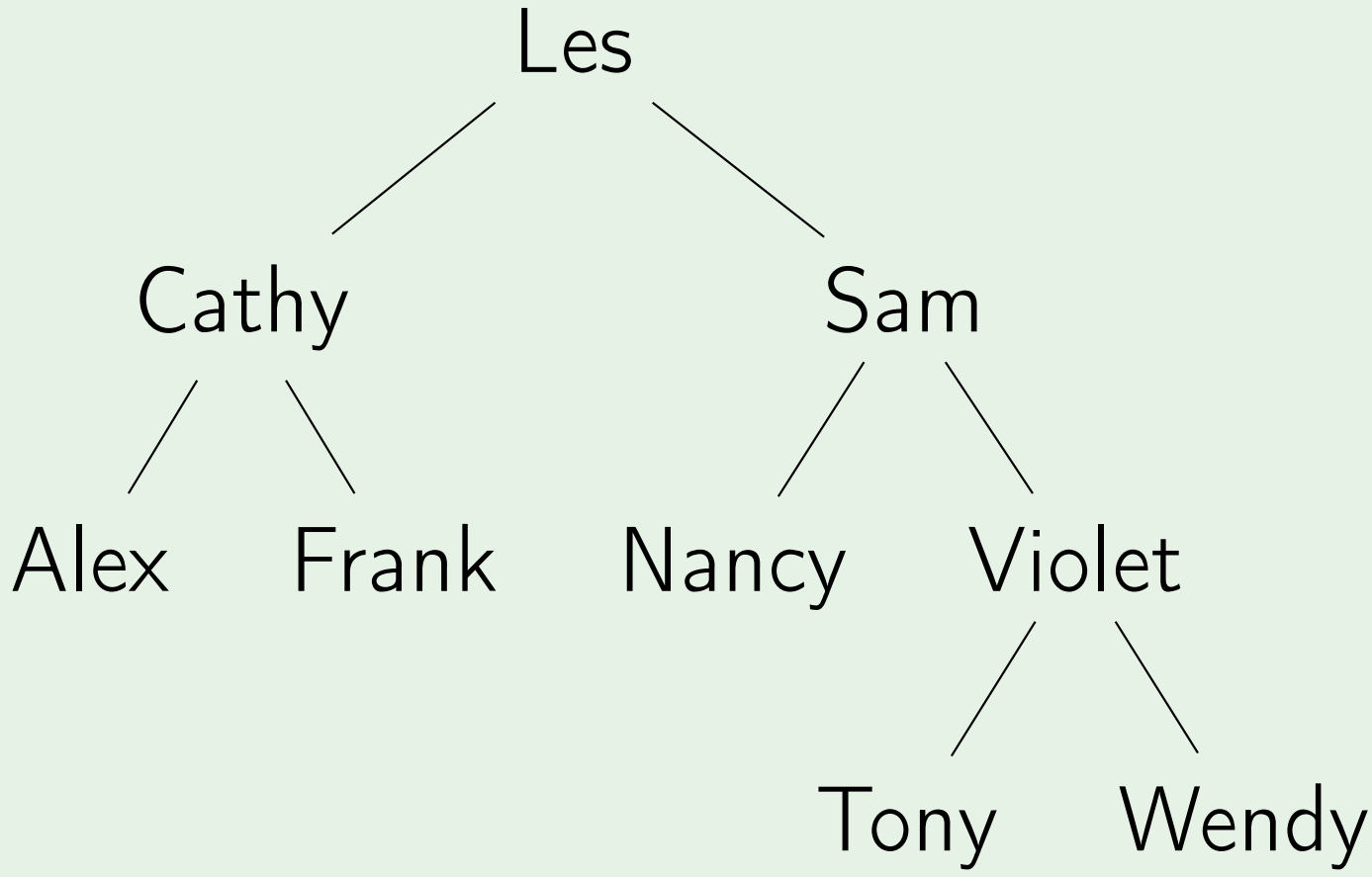
Often we want to visit the nodes of a tree in a particular order.
For example, print the nodes of the tree.

- Depth-first: We completely traverse one sub-tree before exploring a sibling sub-tree.
- Breadth-first: We traverse all nodes at one level before progressing to the next level.

Depth-first

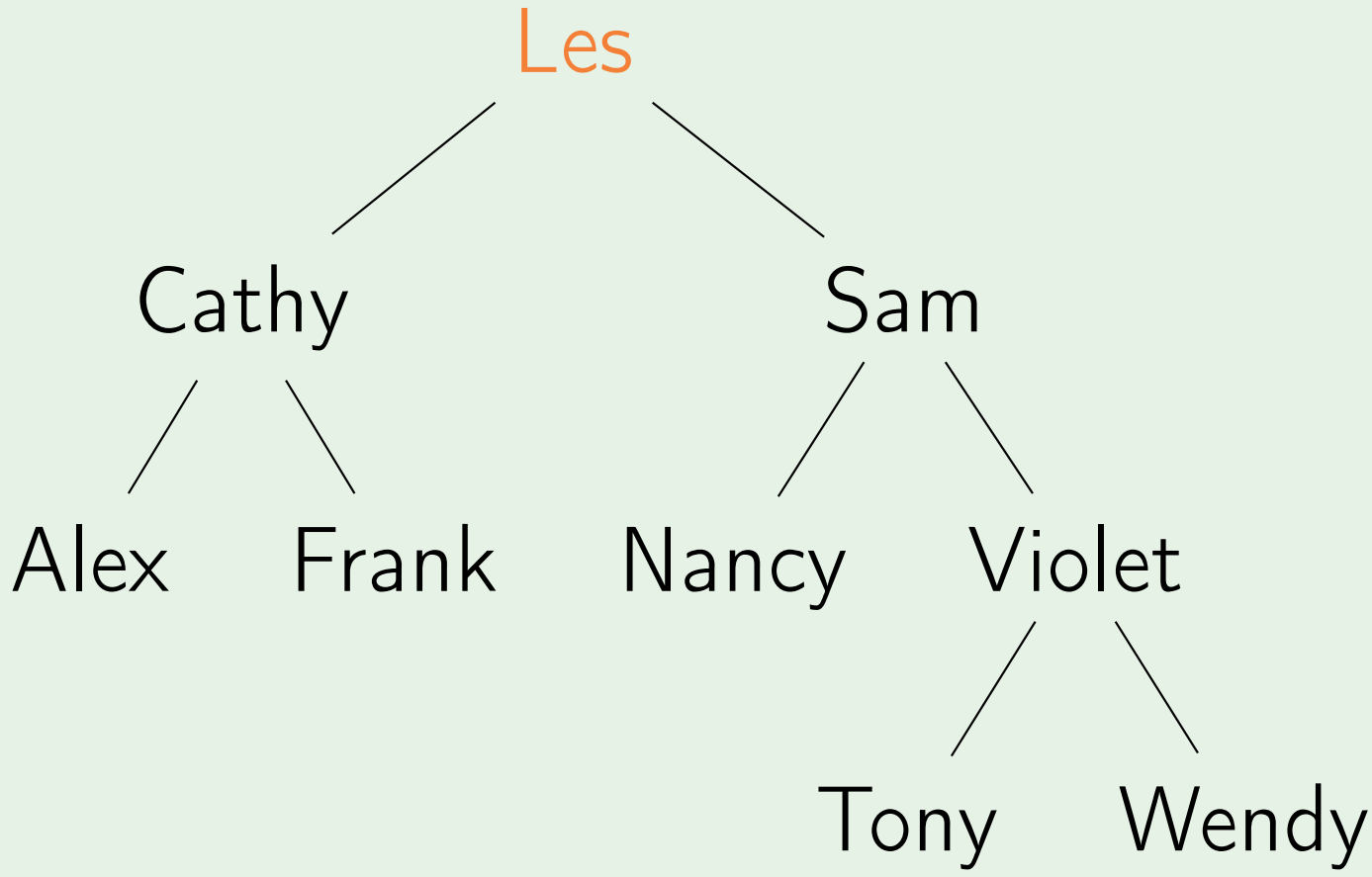
```
InOrderTraversal(tree)
if tree = nil :
   return
InOrderTraversal(tree.left)
Print(tree.key)
InOrderTraversal(tree.right)
```

InOrderTraversal



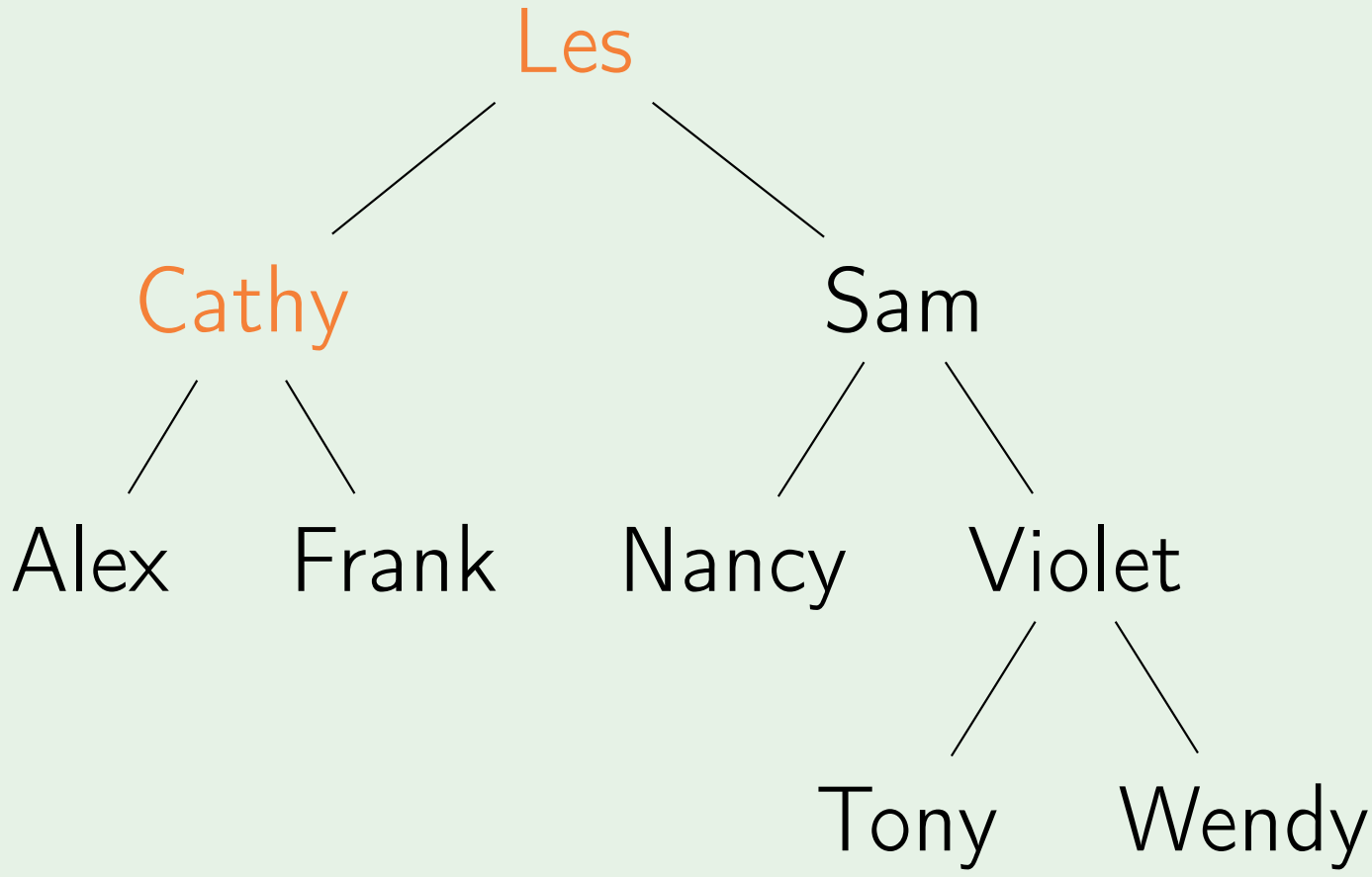
Output:

InOrderTraversal



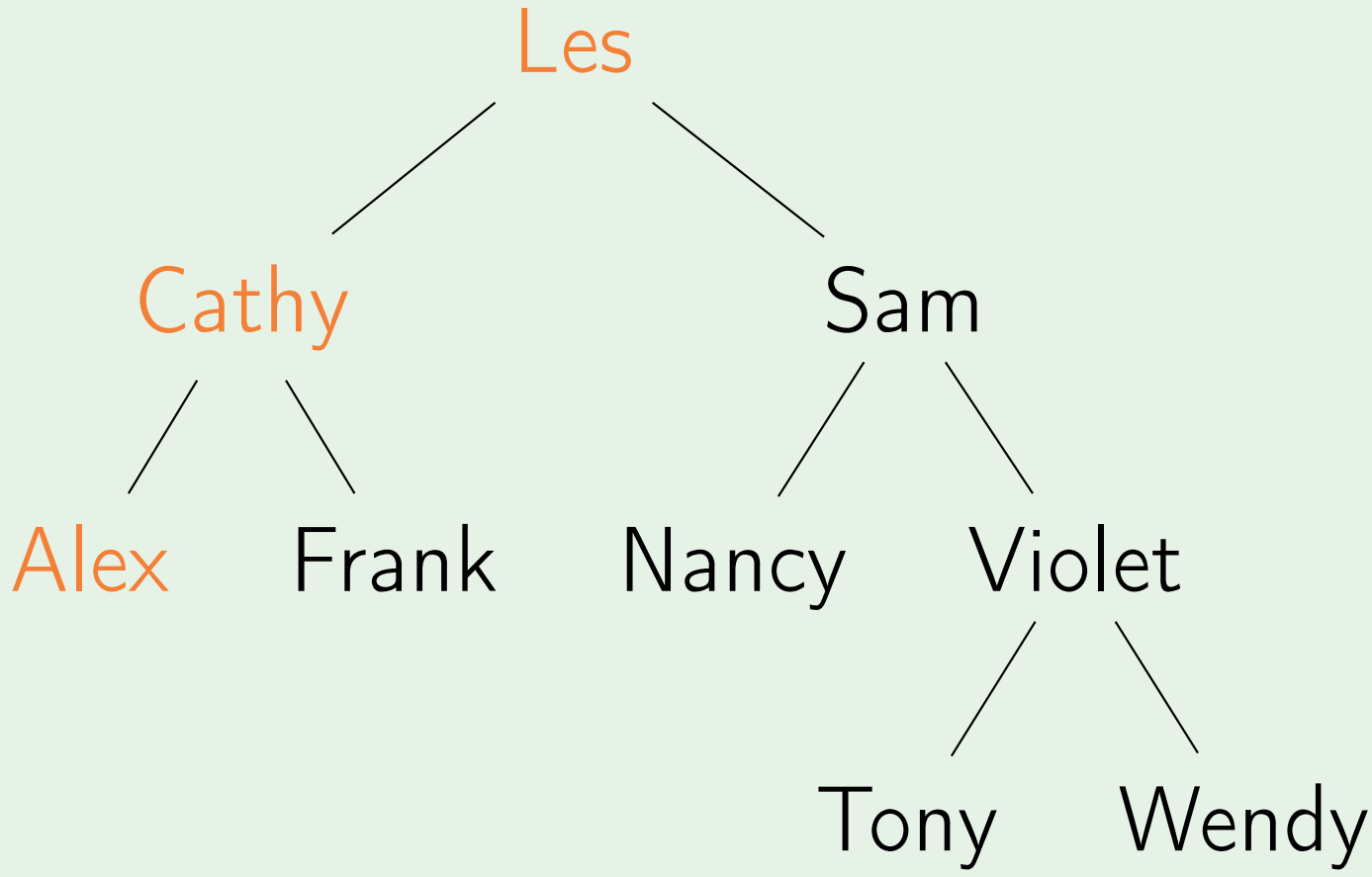
Output:

InOrderTraversal



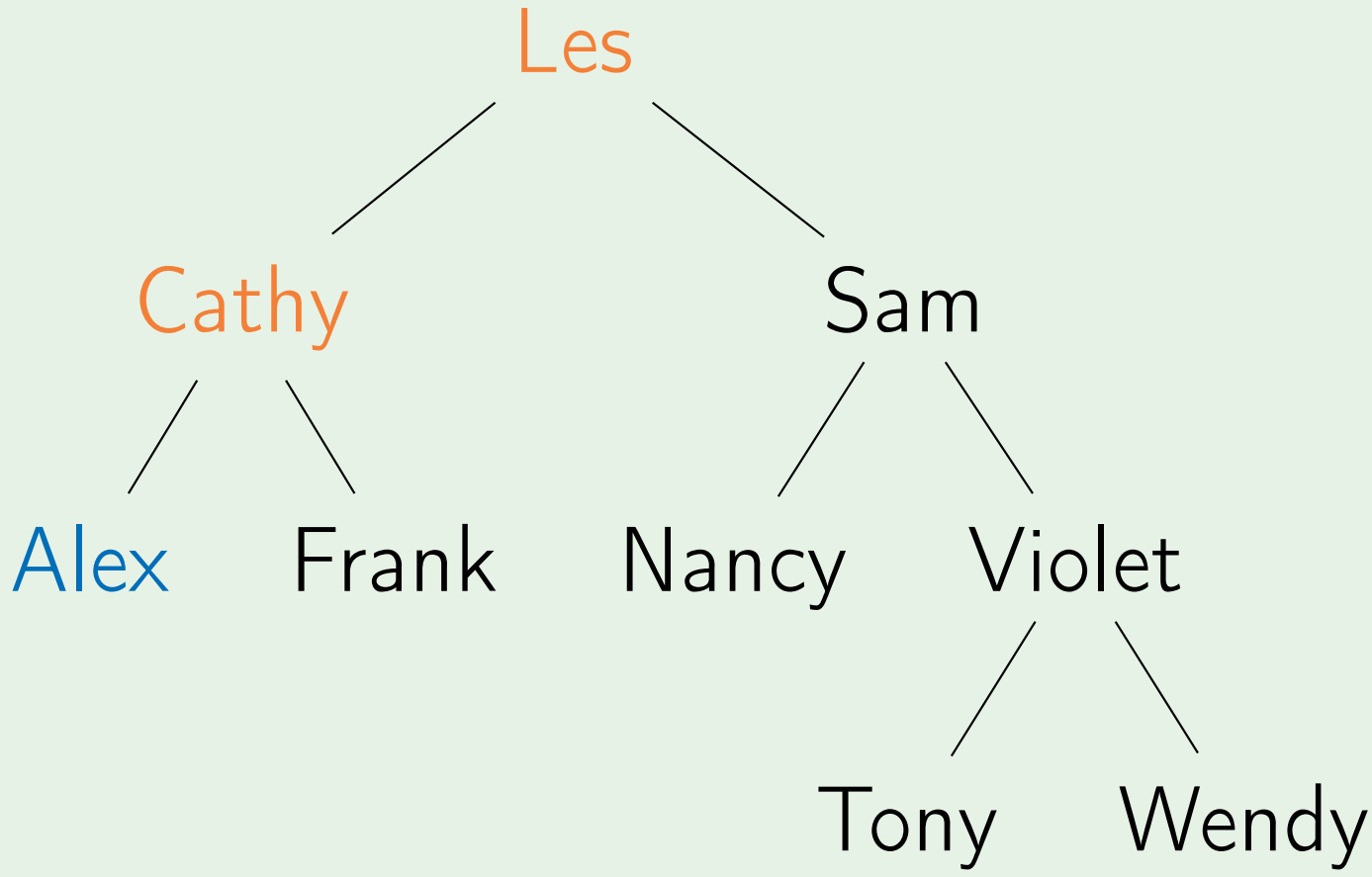
Output:

InOrderTraversal



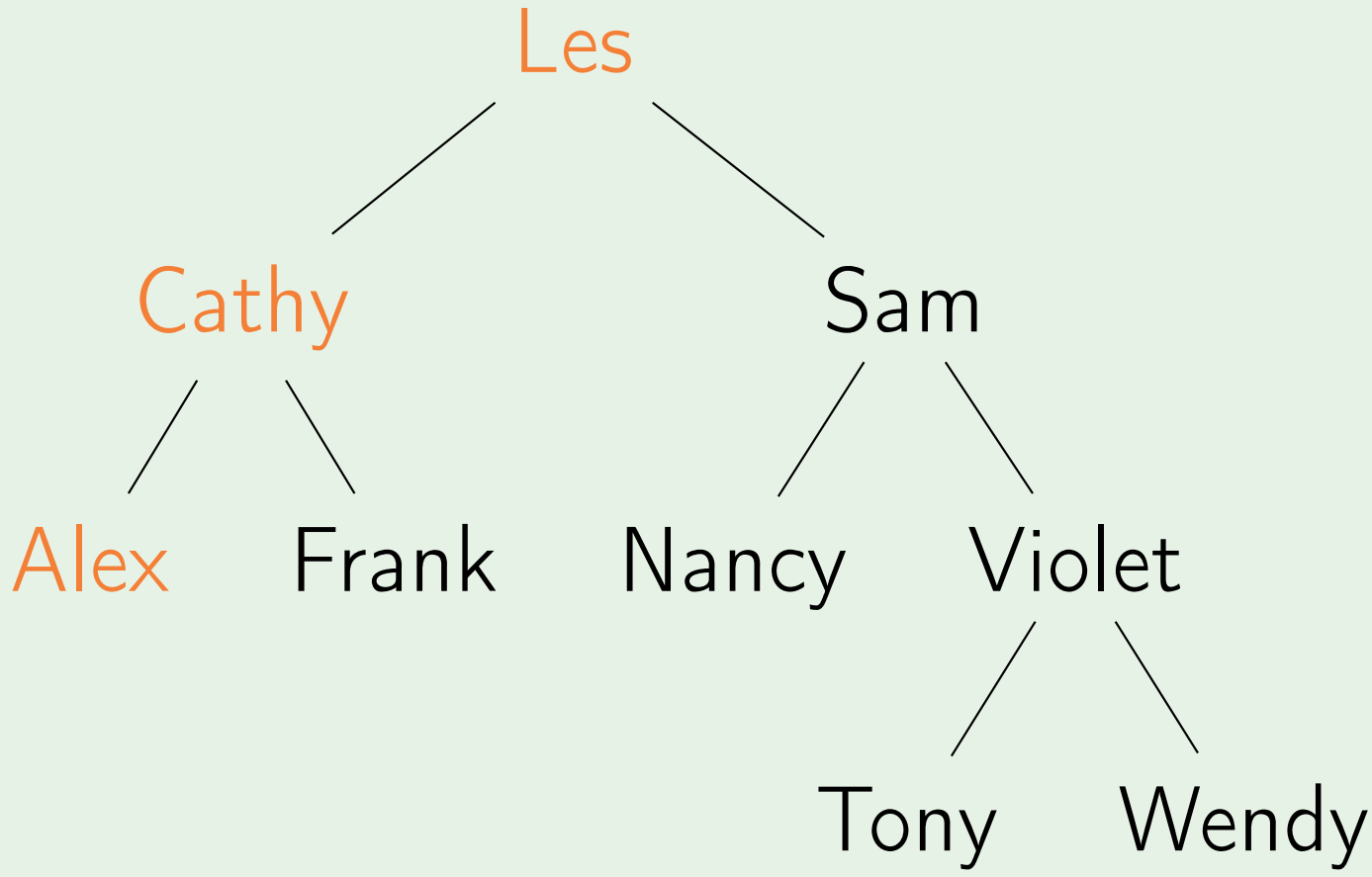
Output:

InOrderTraversal



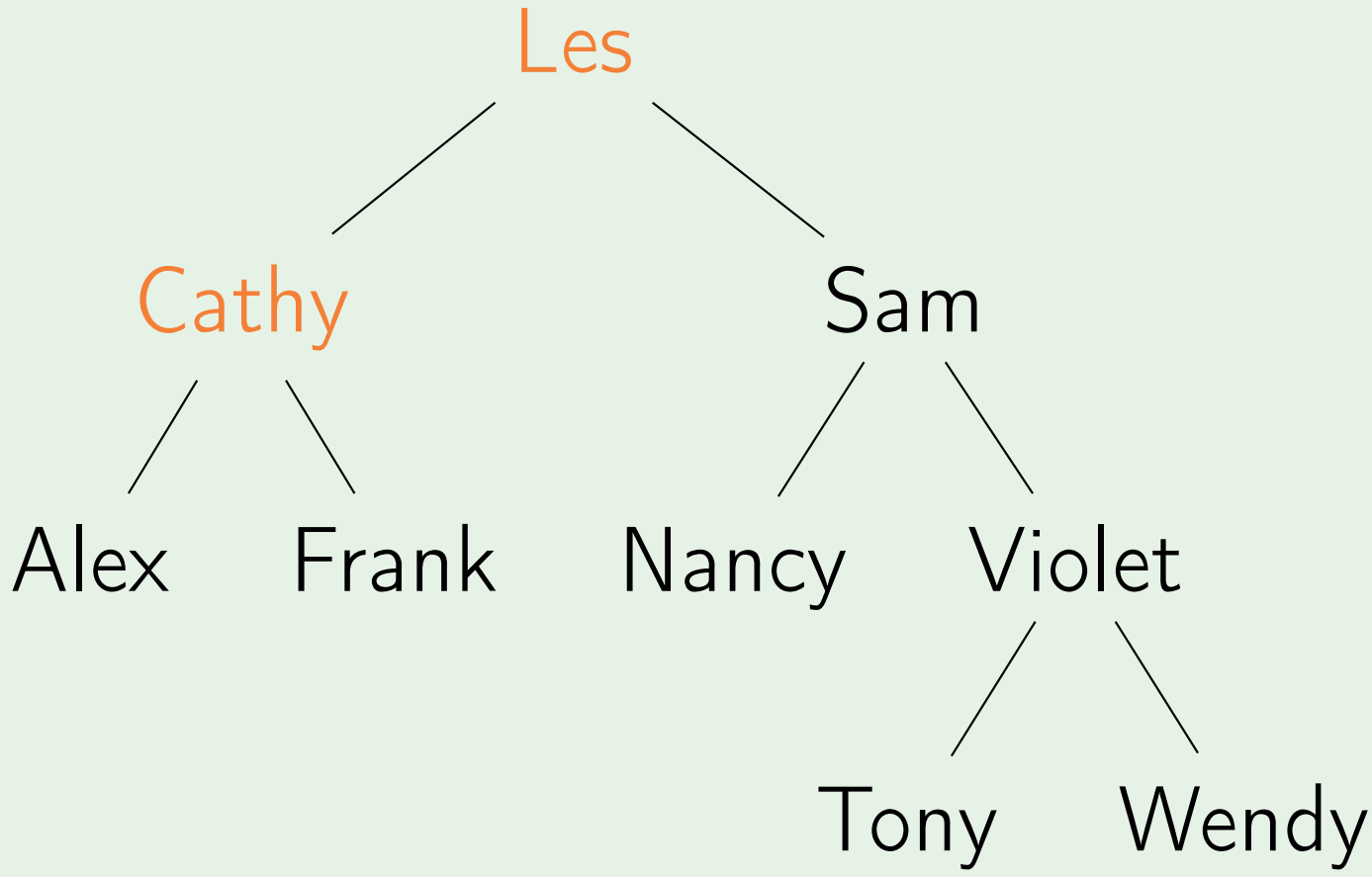
Output: Alex

InOrderTraversal



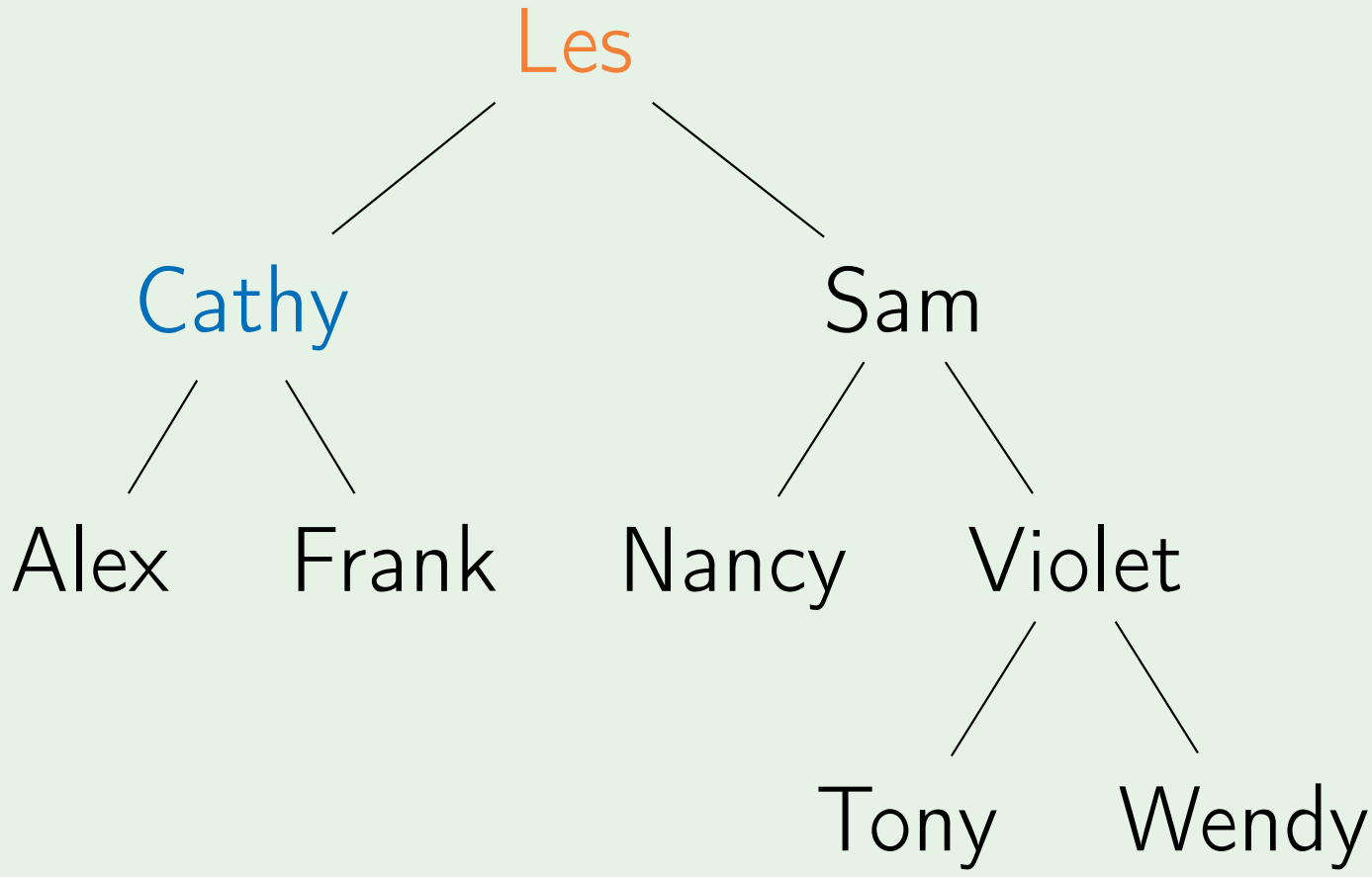
Output: Alex

InOrderTraversal



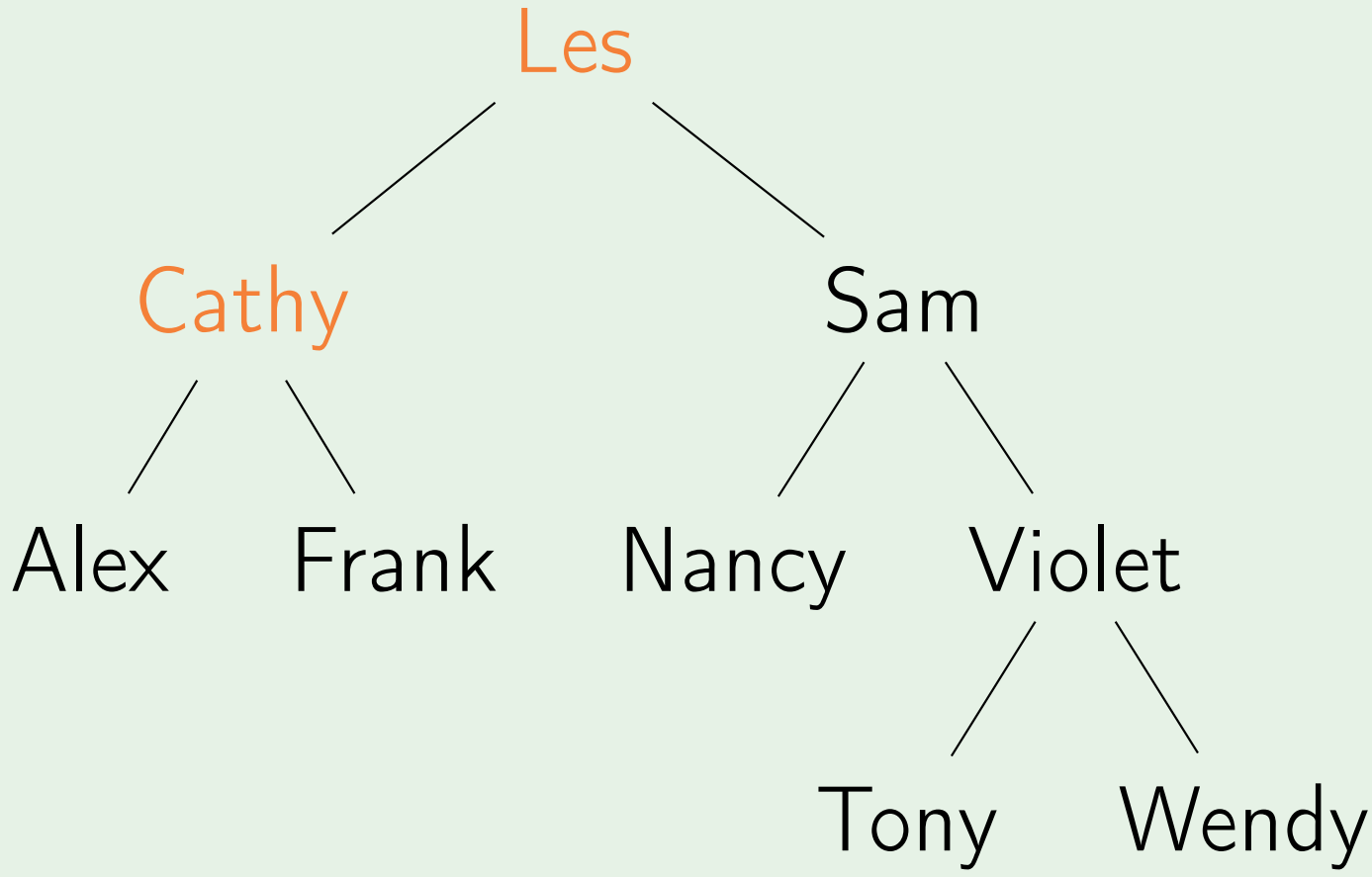
Output: Alex

InOrderTraversal



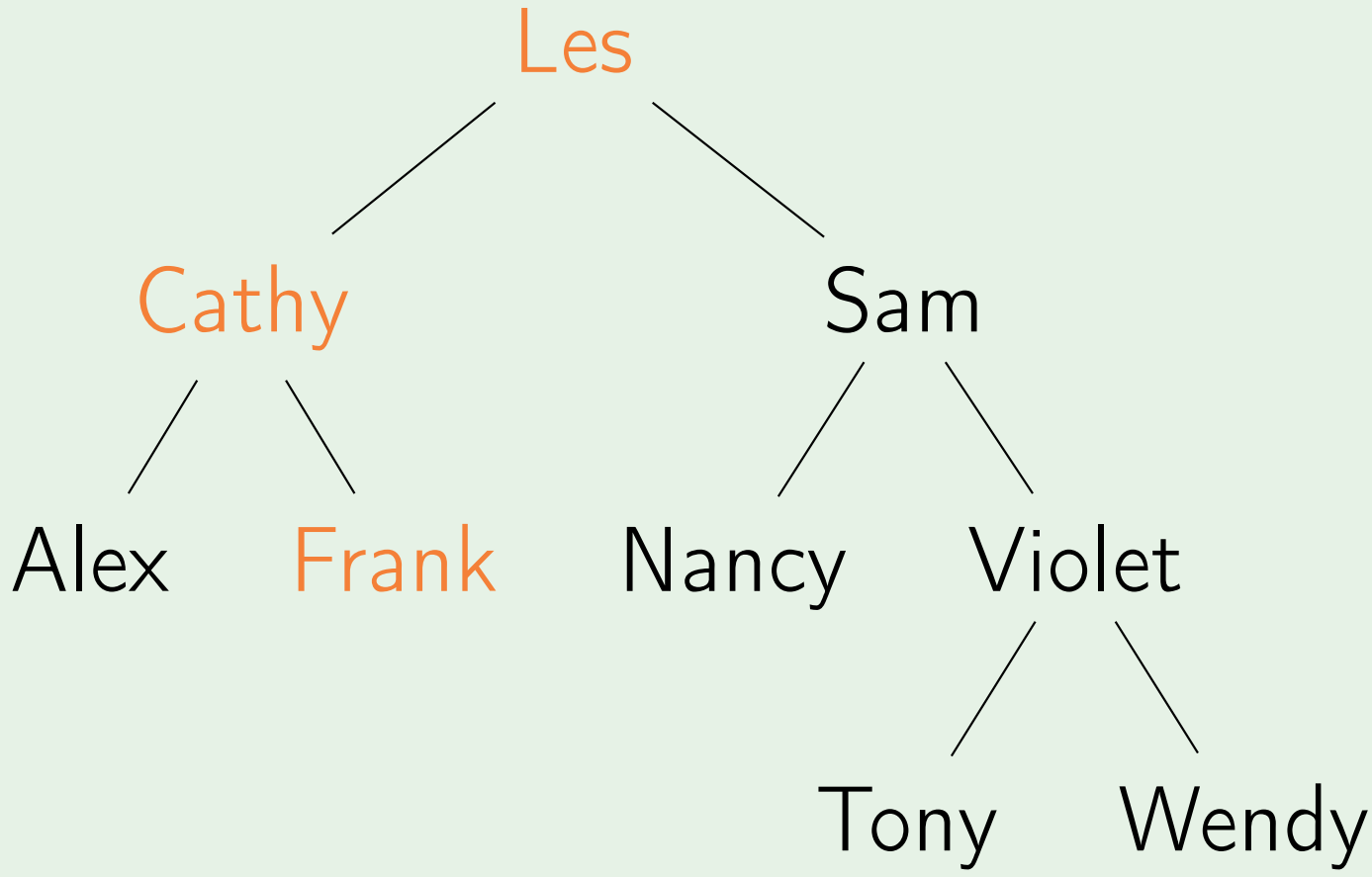
Output: Alex Cathy

InOrderTraversal



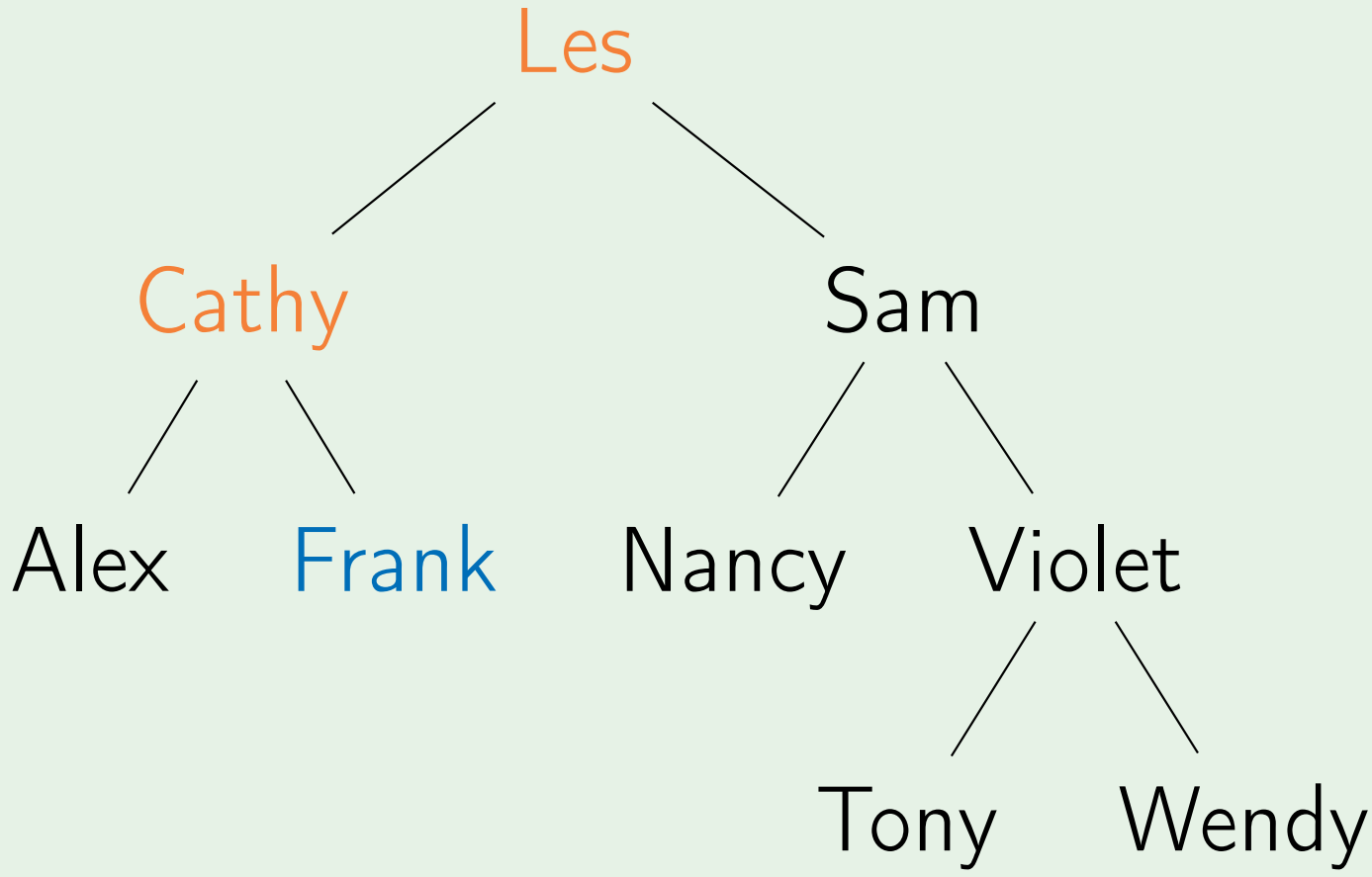
Output: Alex Cathy

InOrderTraversal



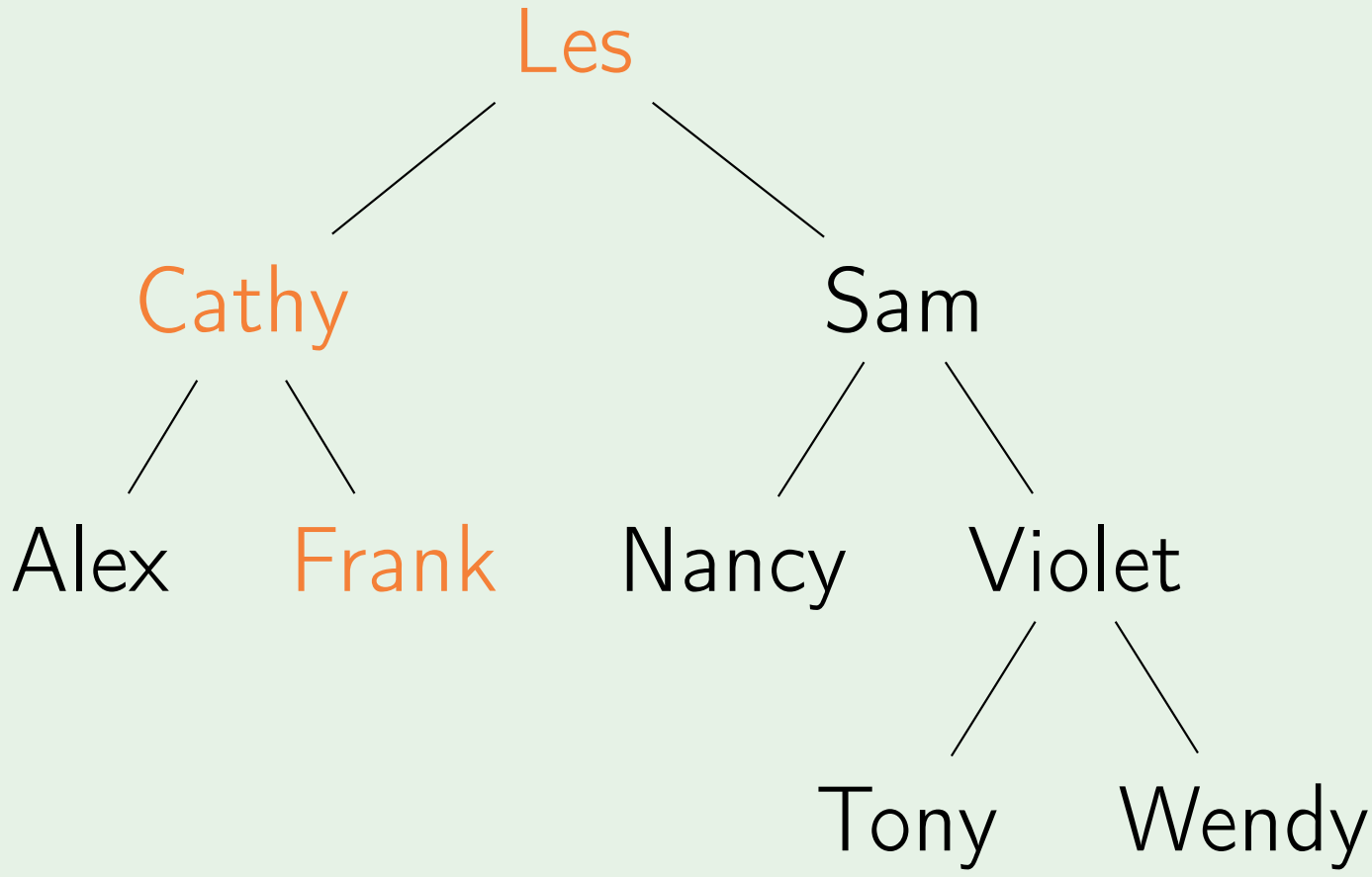
Output: Alex Cathy

InOrderTraversal



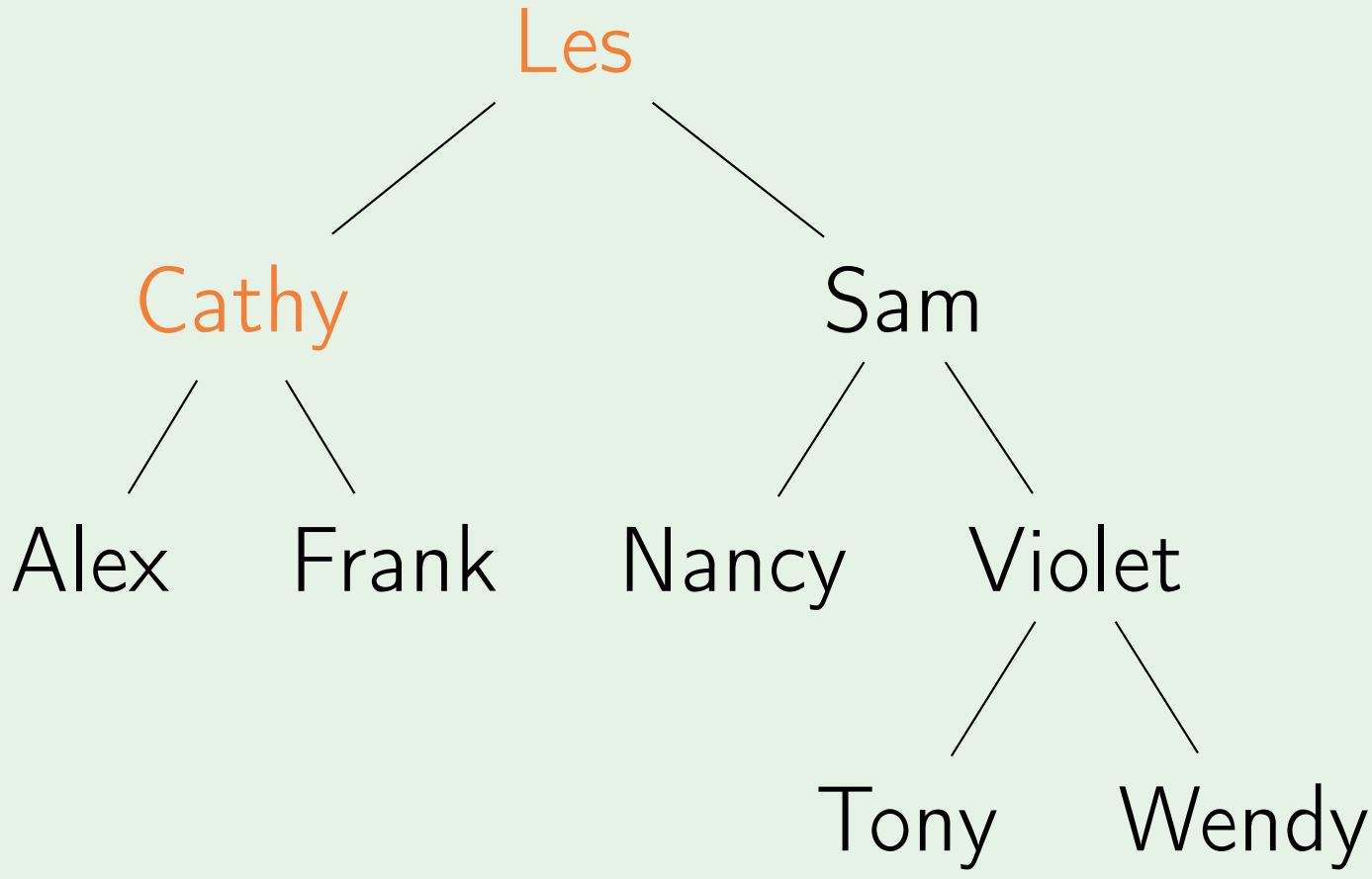
Output: Alex Cathy Frank

InOrderTraversal



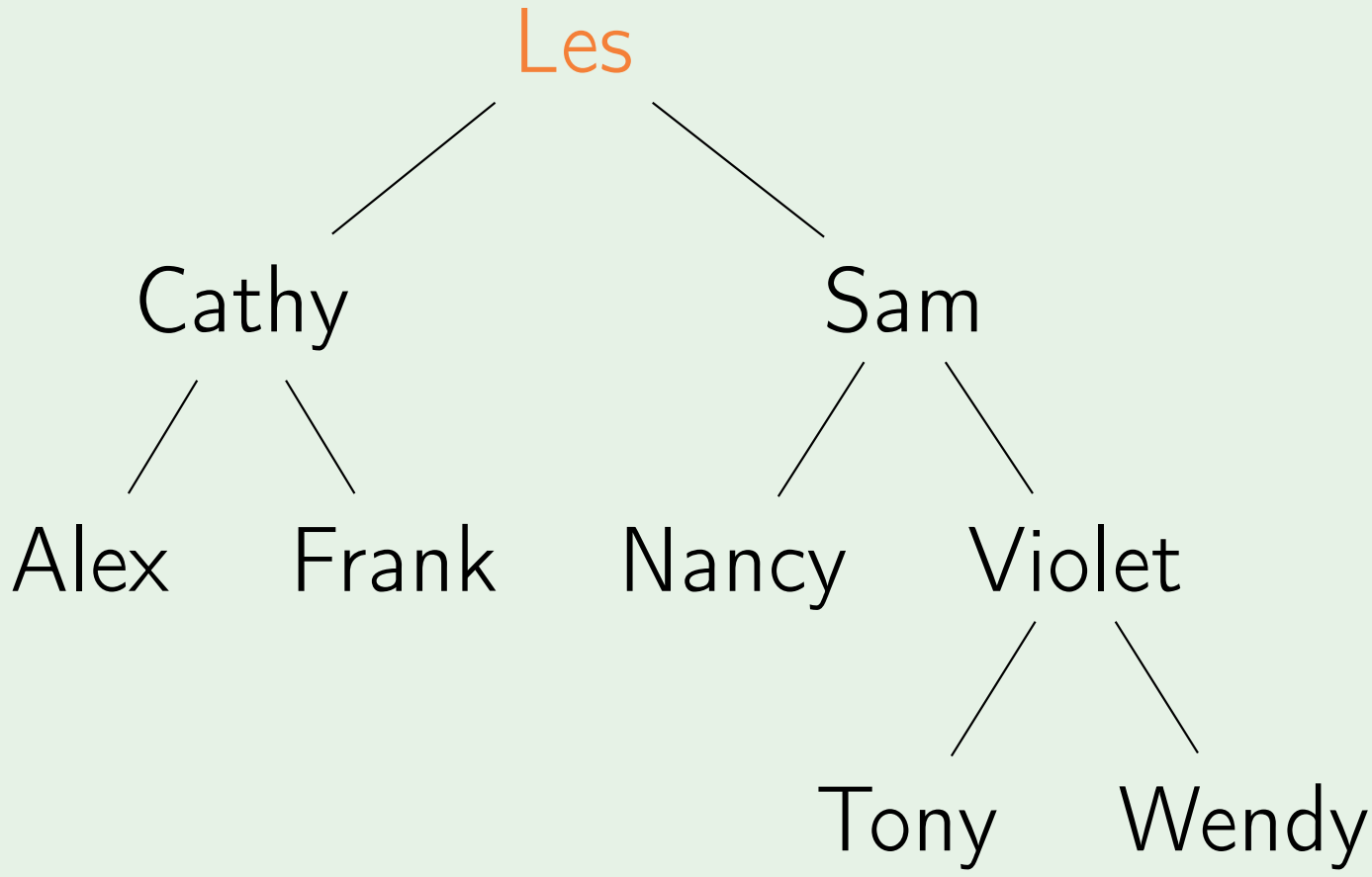
Output: Alex Cathy Frank

InOrderTraversal



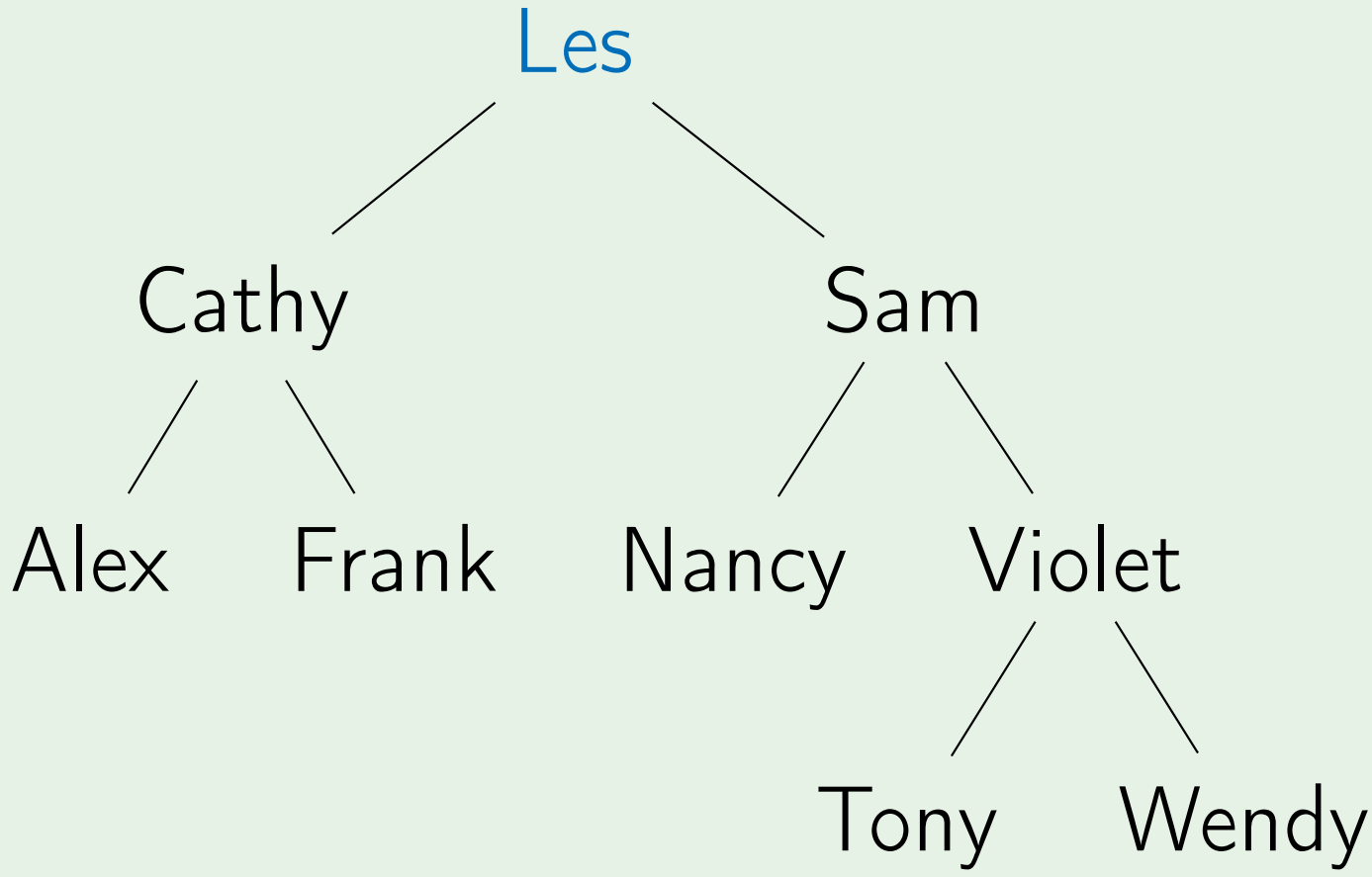
Output: Alex Cathy Frank

InOrderTraversal



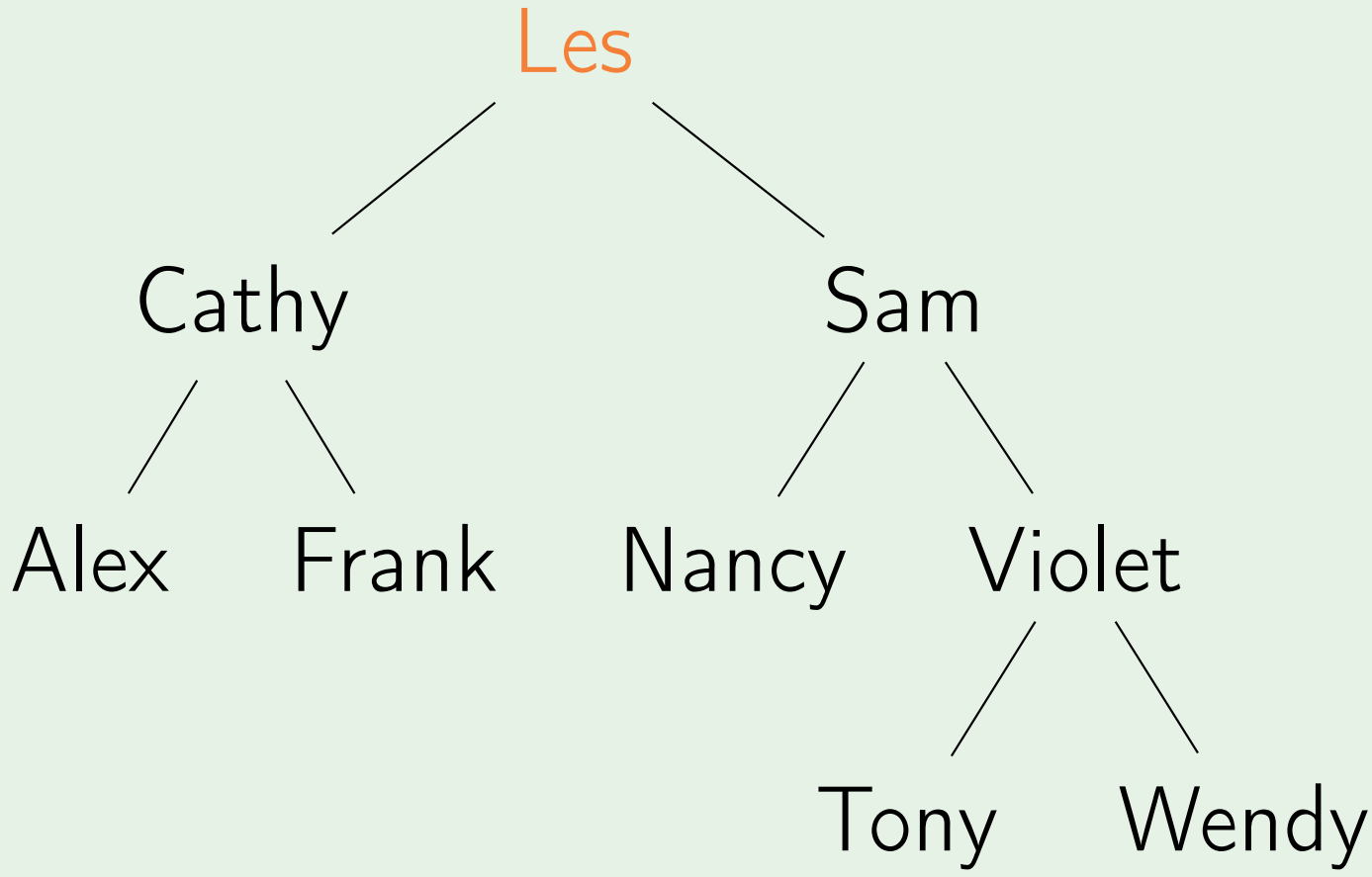
Output: Alex Cathy Frank

InOrderTraversal



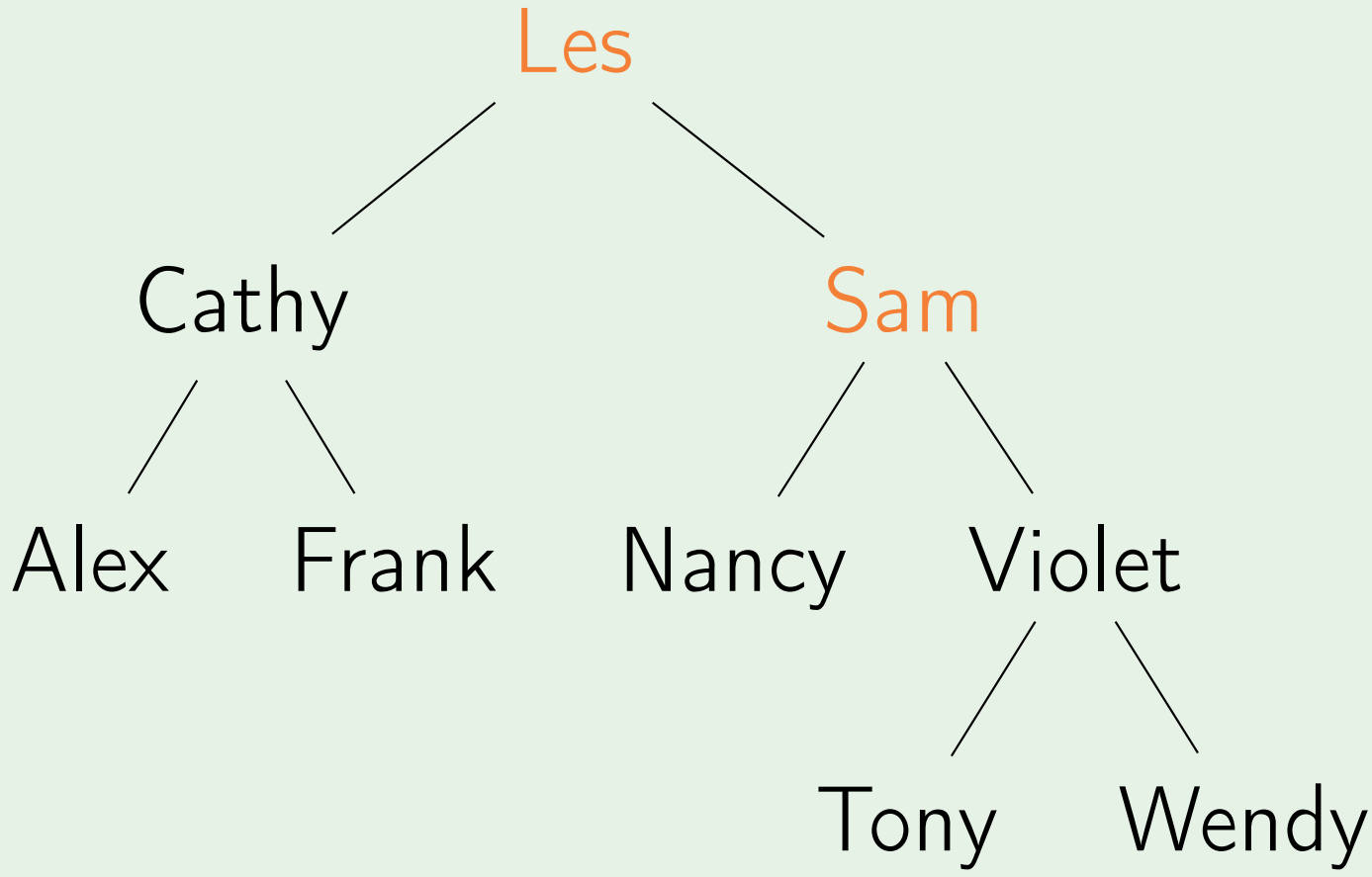
Output: Alex Cathy Frank Les

InOrderTraversal



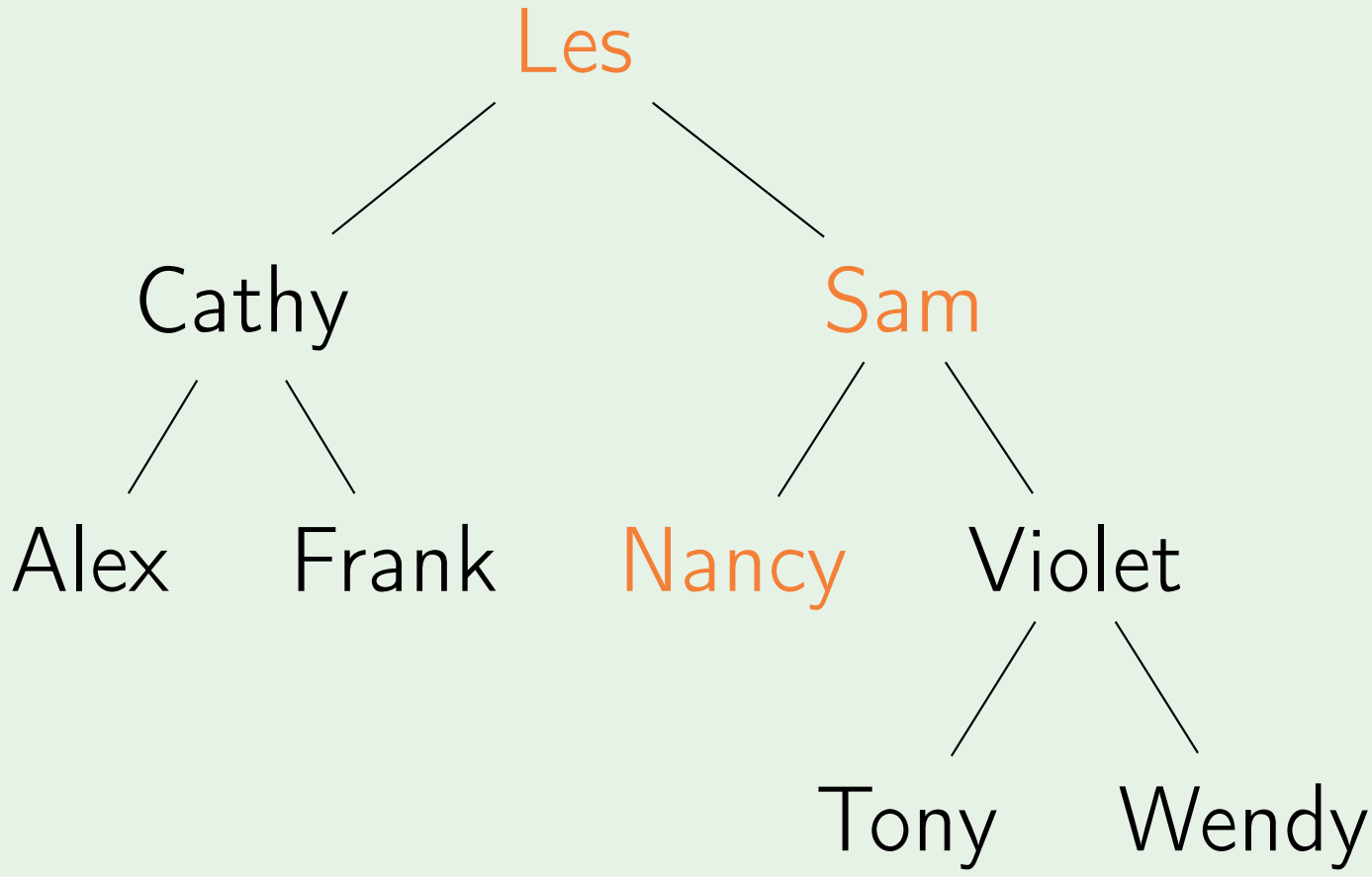
Output: Alex Cathy Frank Les

InOrderTraversal



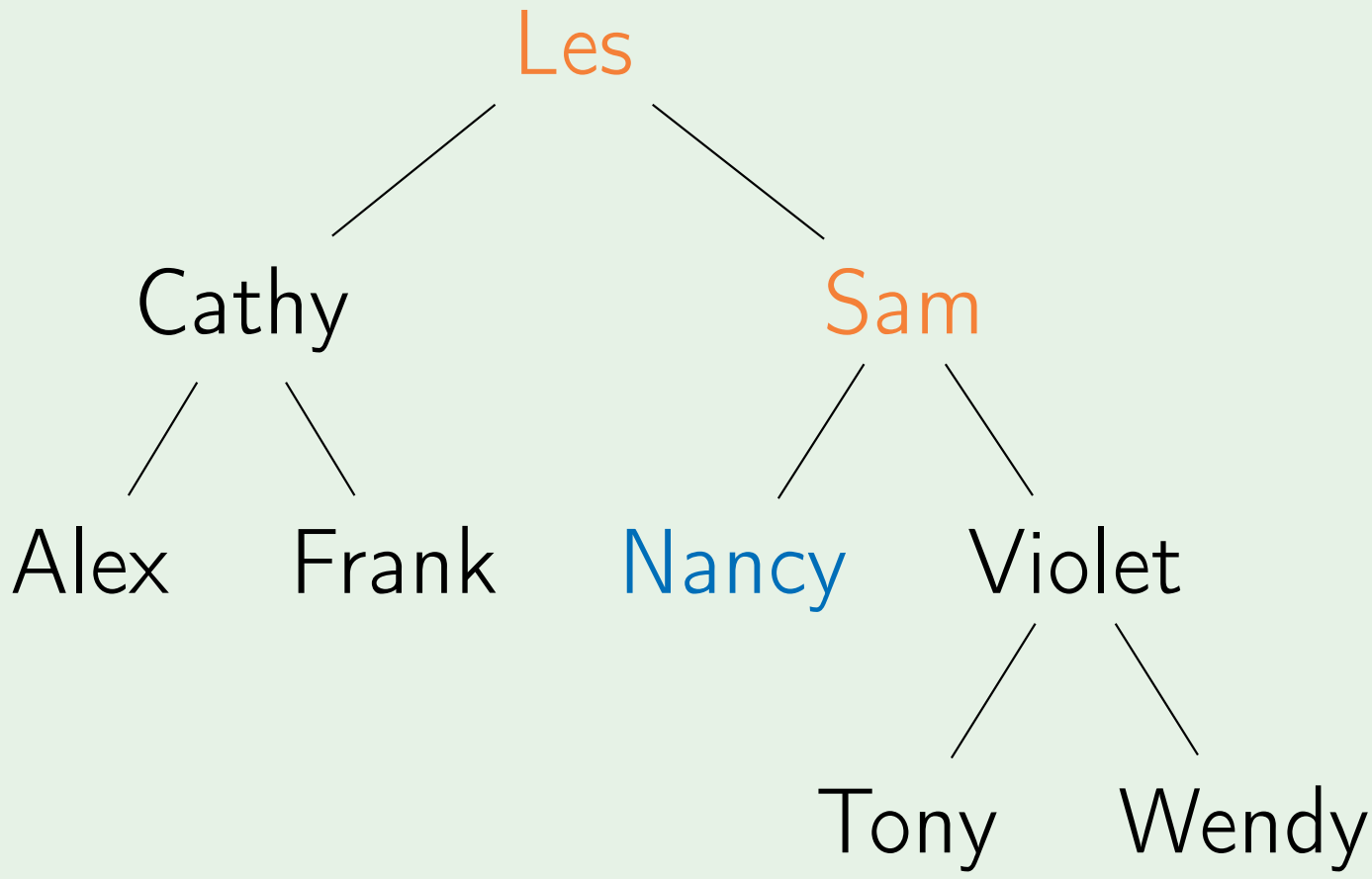
Output: Alex Cathy Frank Les

InOrderTraversal



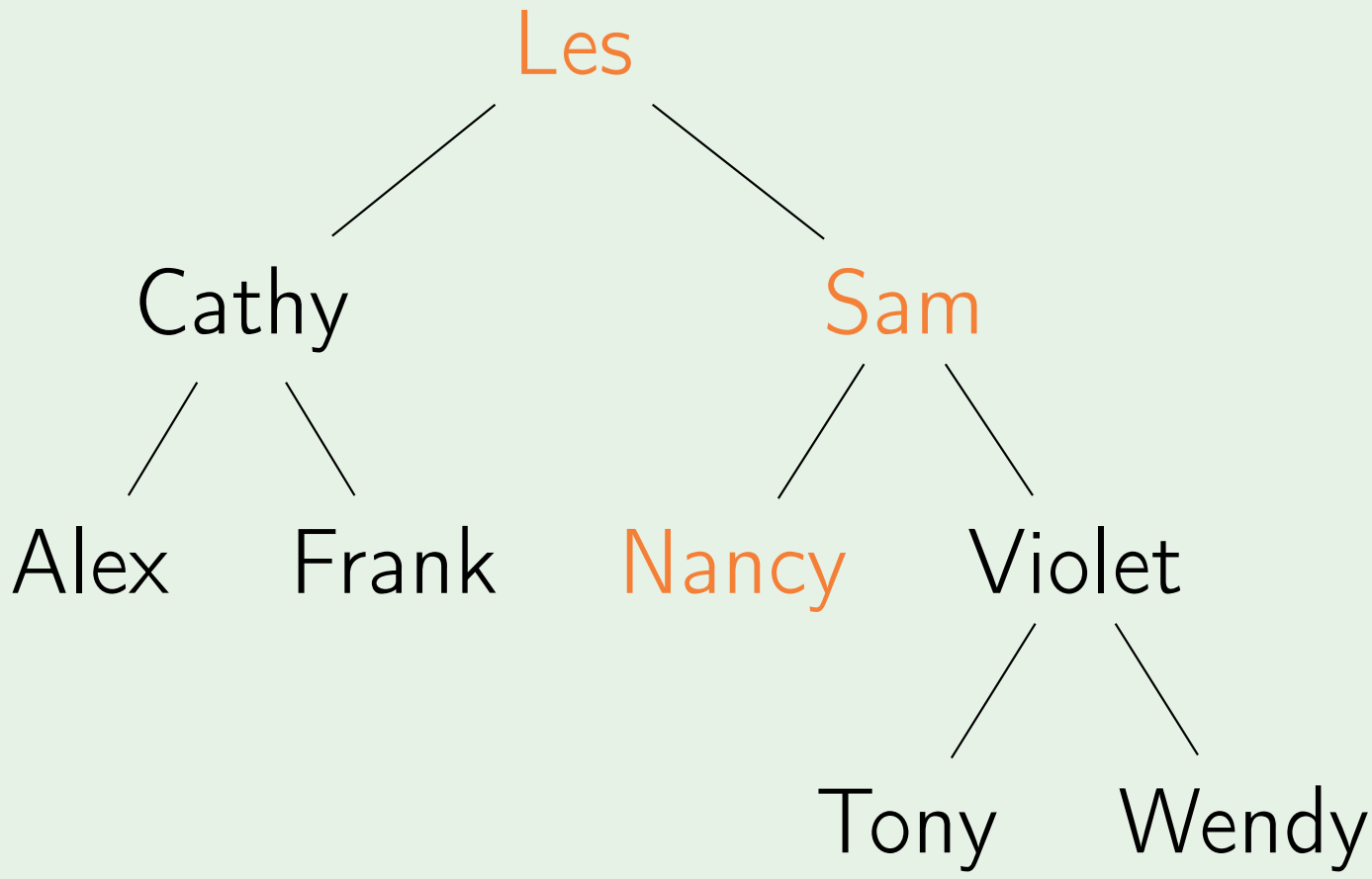
Output: Alex Cathy Frank Les

InOrderTraversal



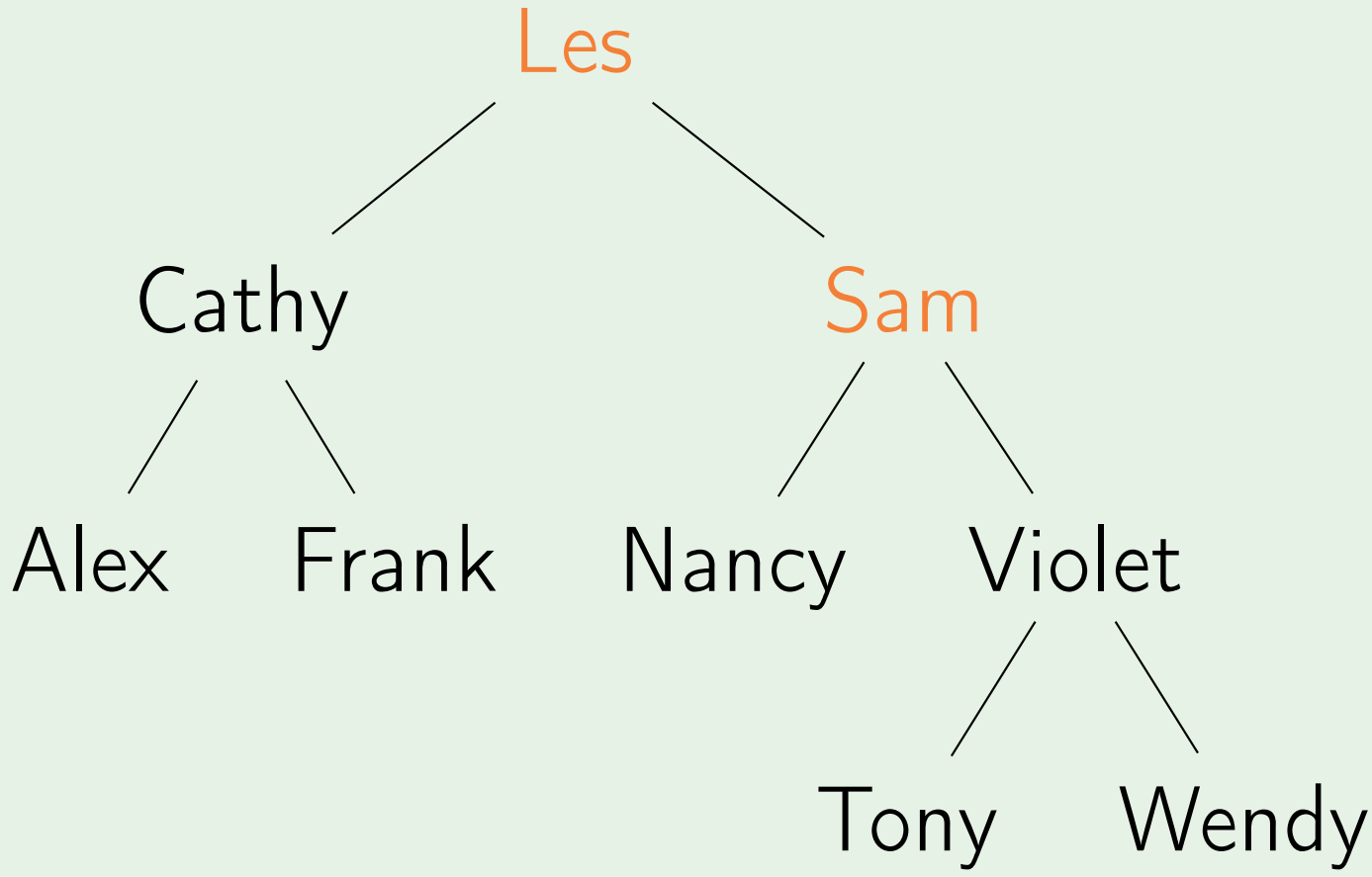
Output: Alex Cathy Frank Les Nancy

InOrderTraversal



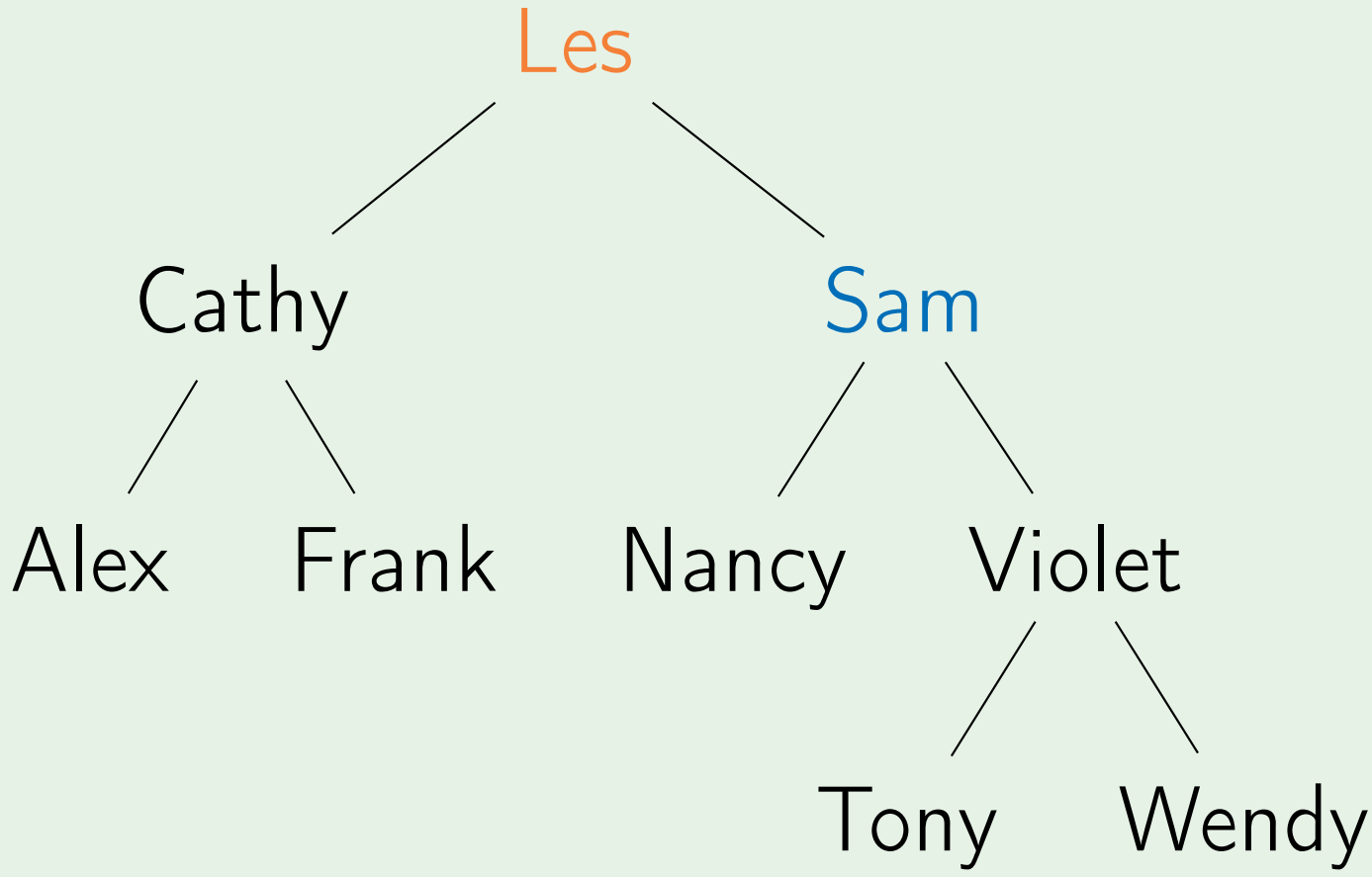
Output: Alex Cathy Frank Les Nancy

InOrderTraversal



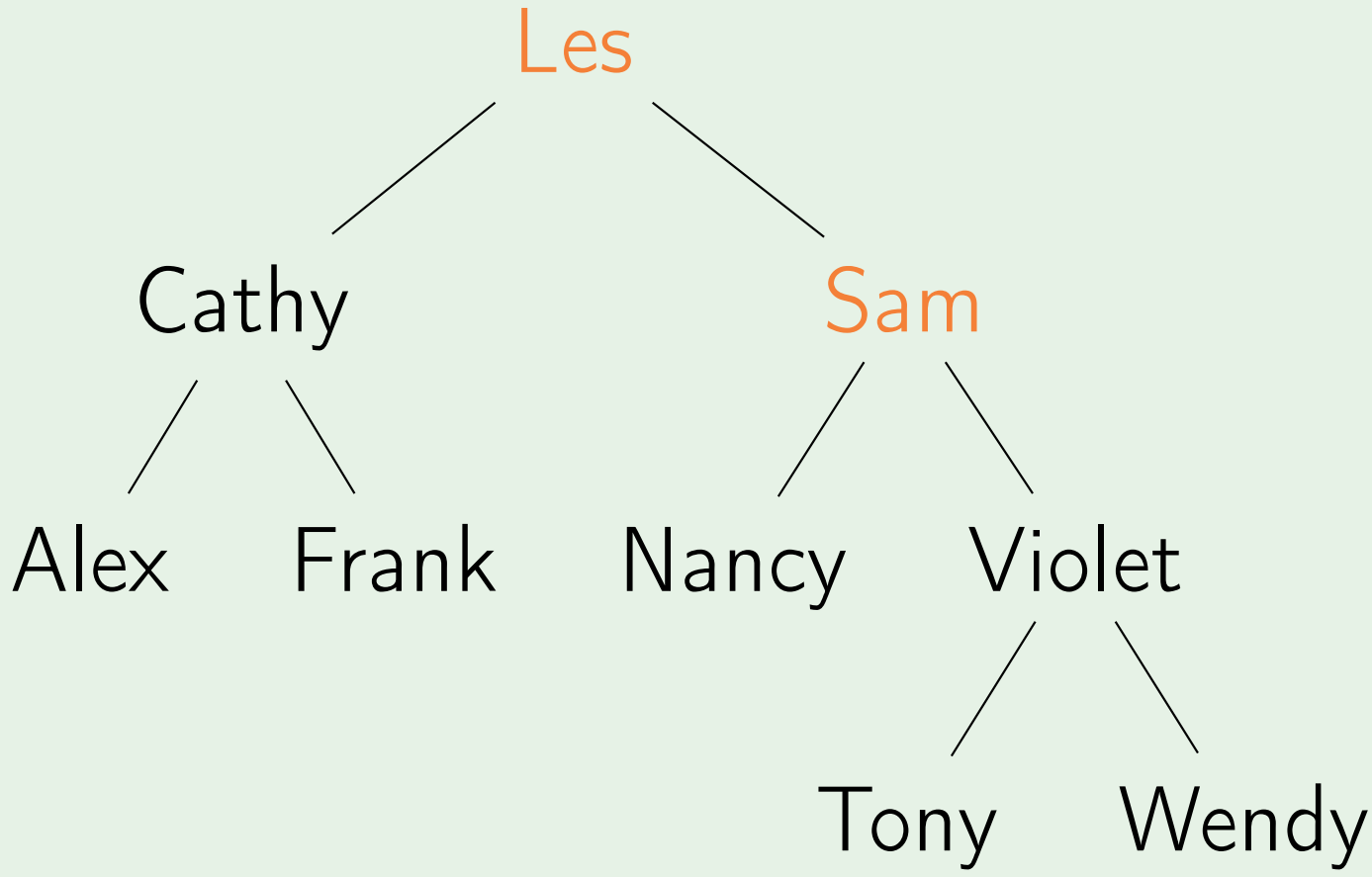
Output: Alex Cathy Frank Les Nancy

InOrderTraversal



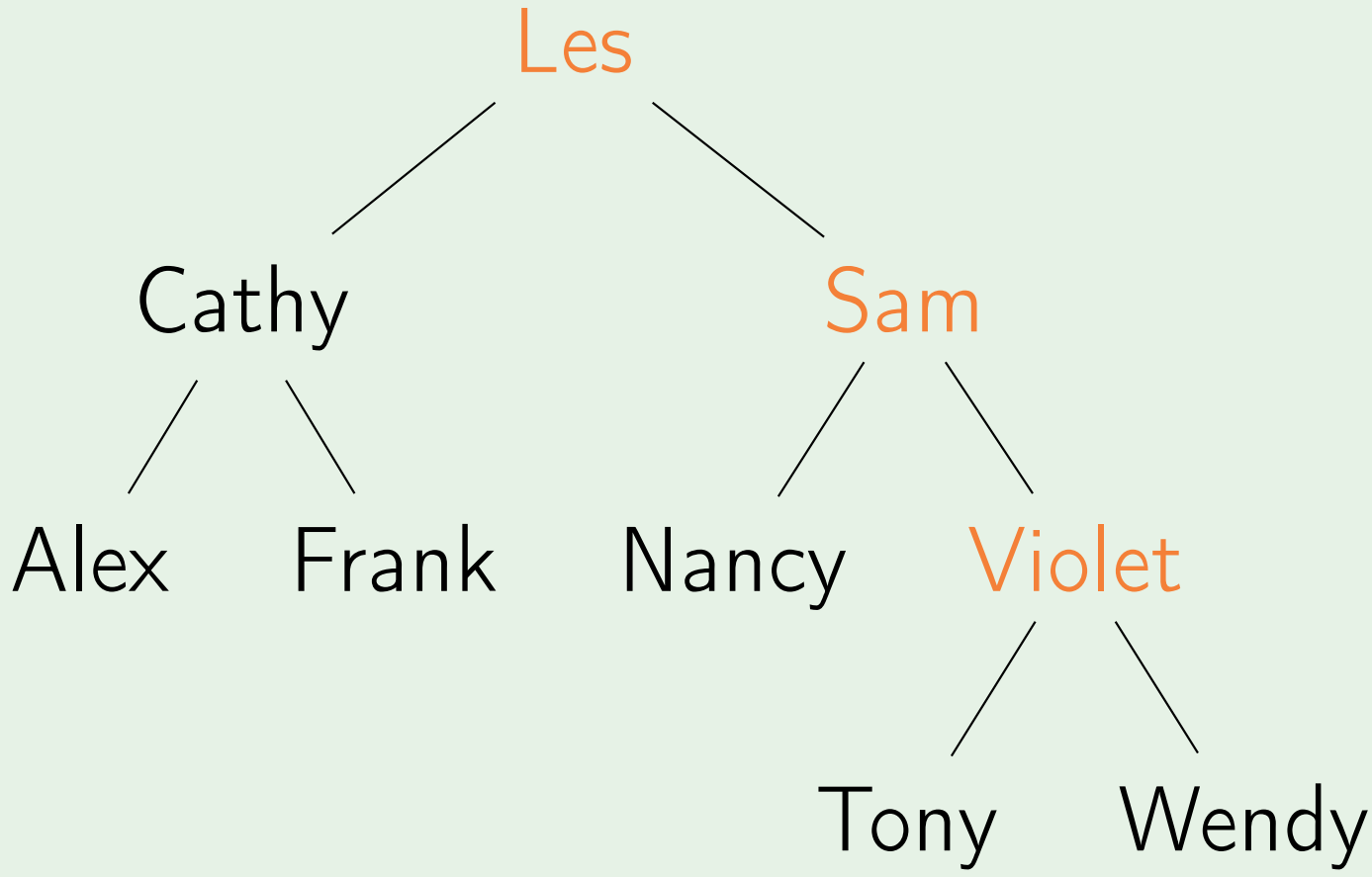
Output: Alex Cathy Frank Les Nancy Sam

InOrderTraversal



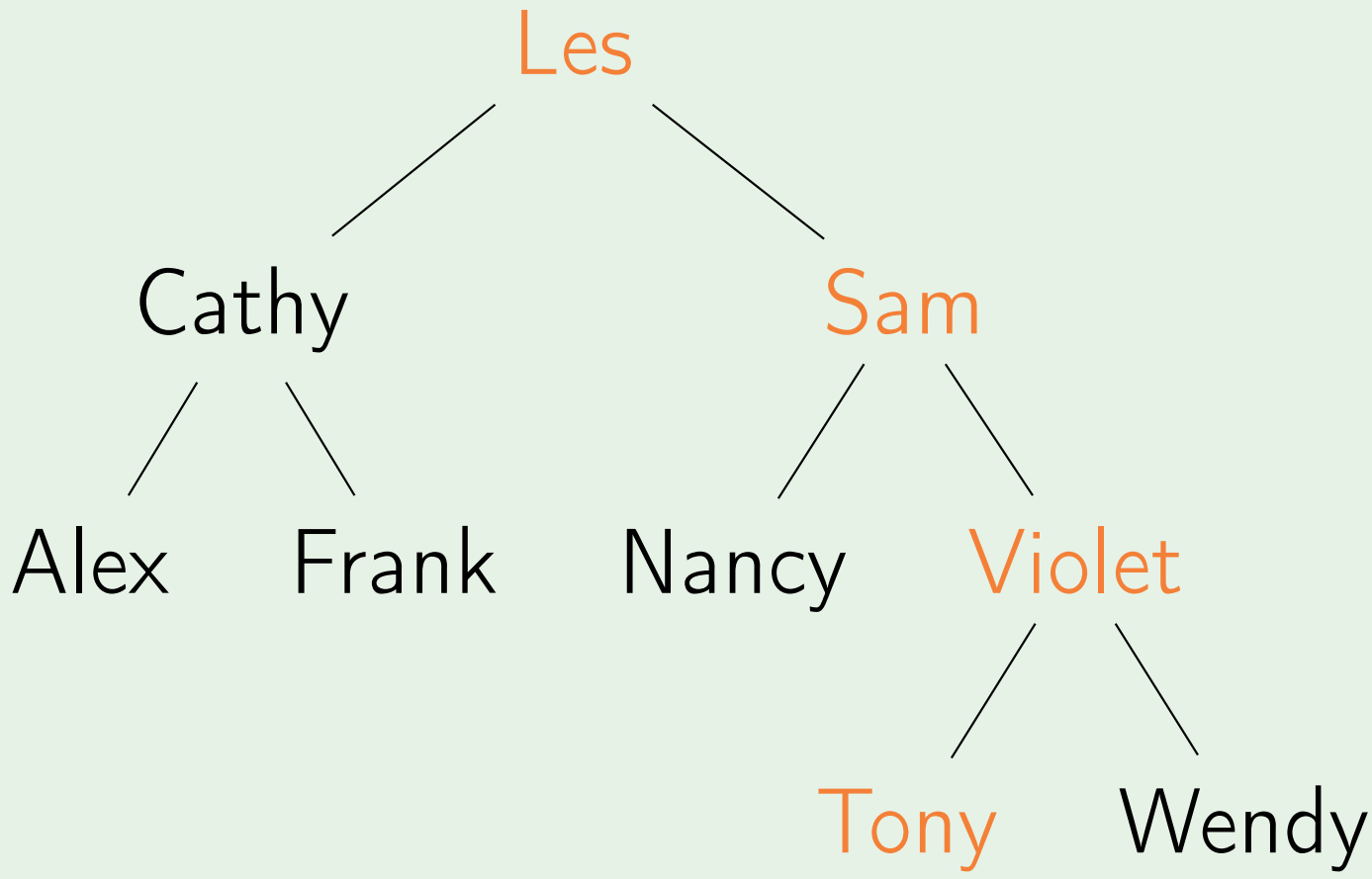
Output: Alex Cathy Frank Les Nancy Sam

InOrderTraversal



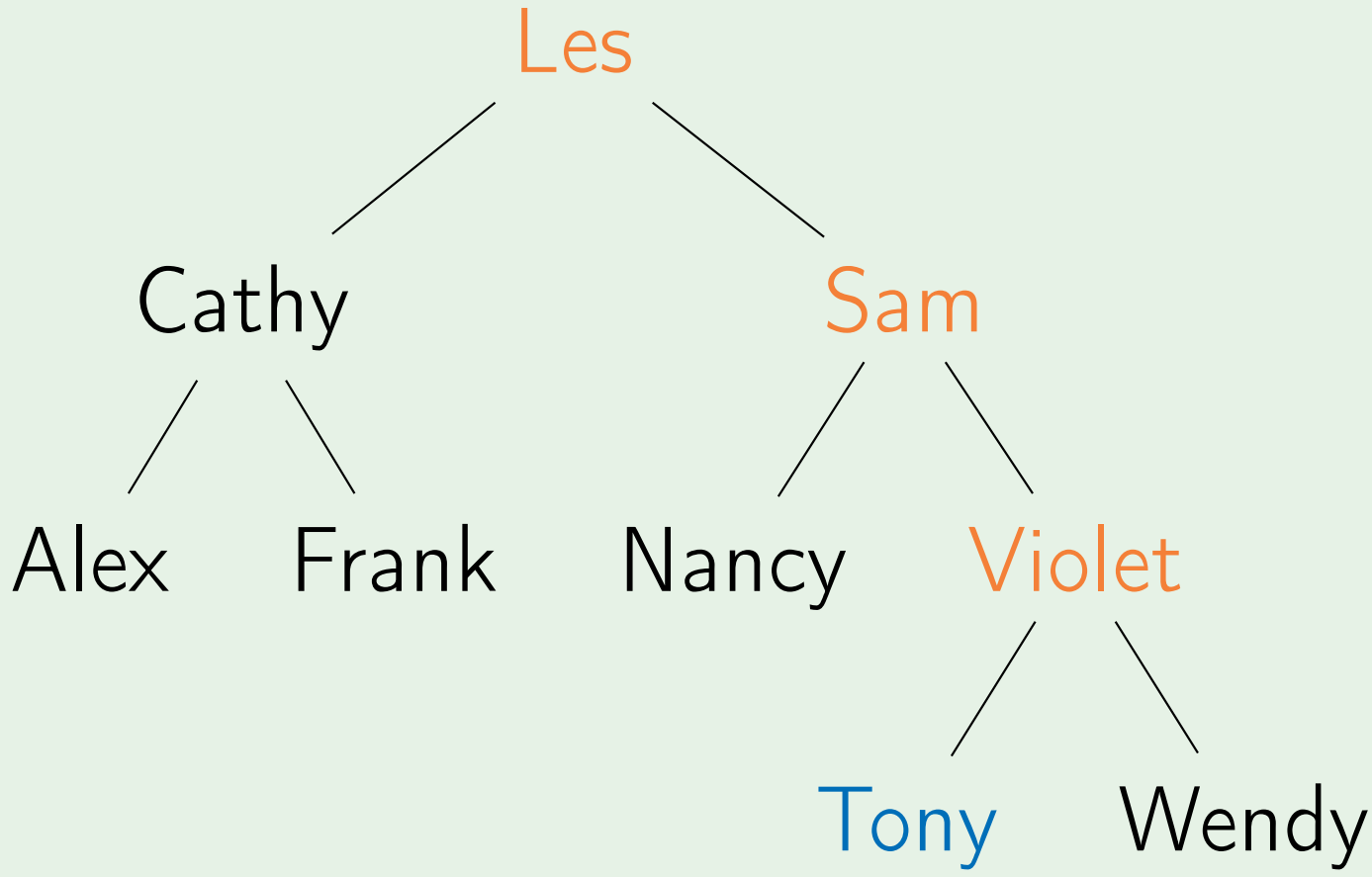
Output: Alex Cathy Frank Les Nancy Sam

InOrderTraversal



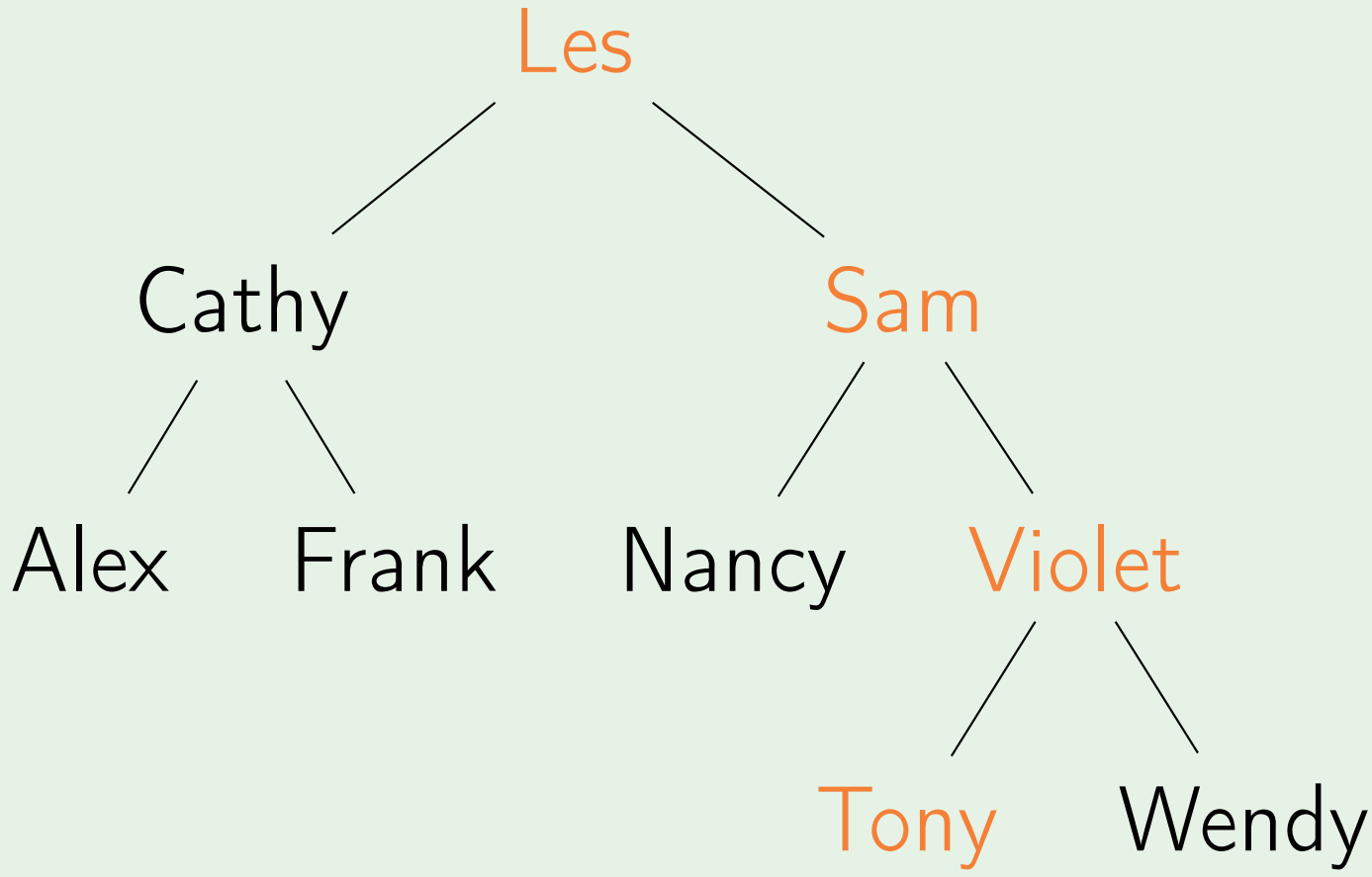
Output: Alex Cathy Frank Les Nancy Sam

InOrderTraversal



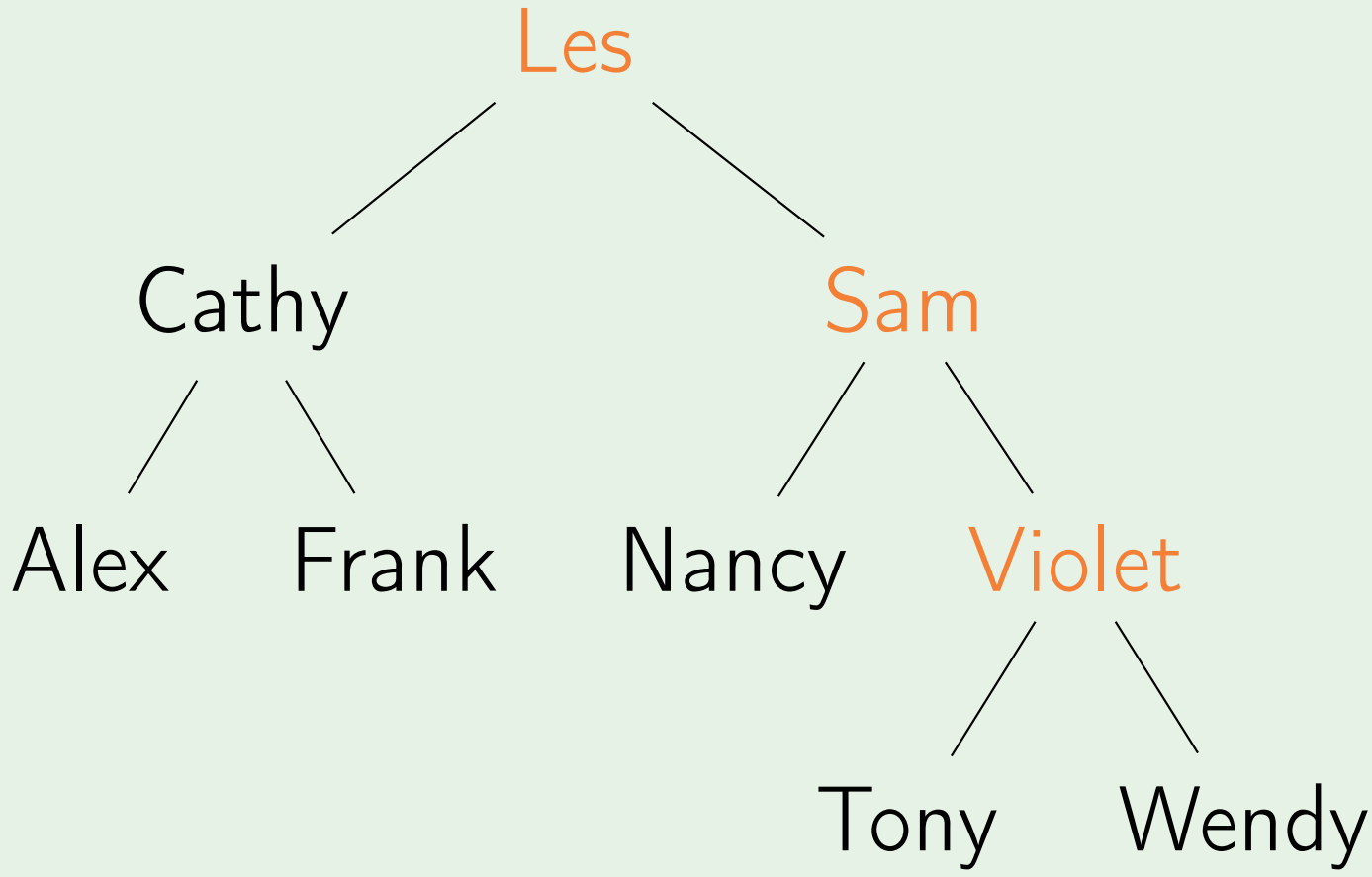
Output: Alex Cathy Frank Les Nancy Sam Tony

InOrderTraversal



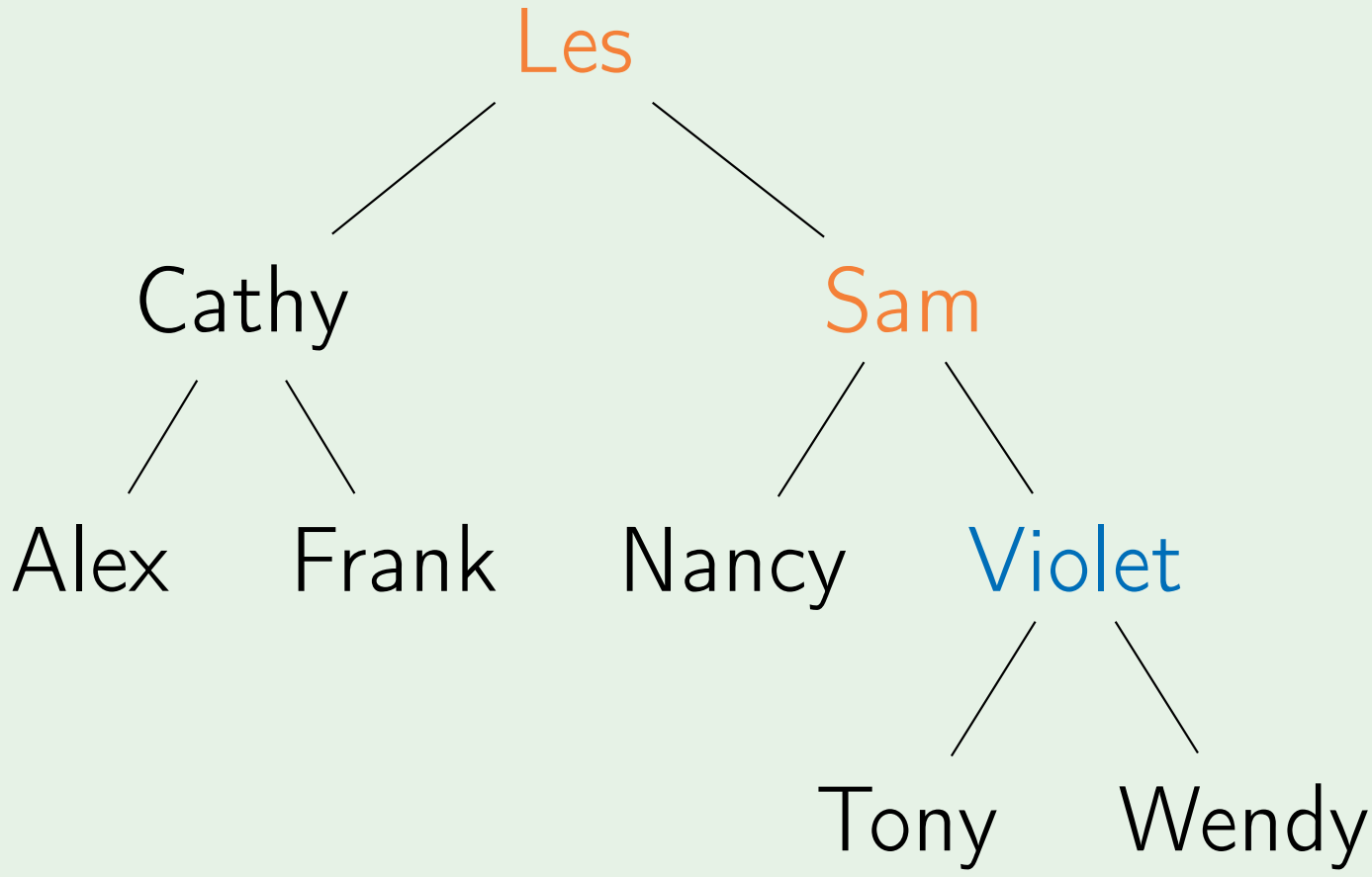
Output: Alex Cathy Frank Les Nancy Sam Tony

InOrderTraversal



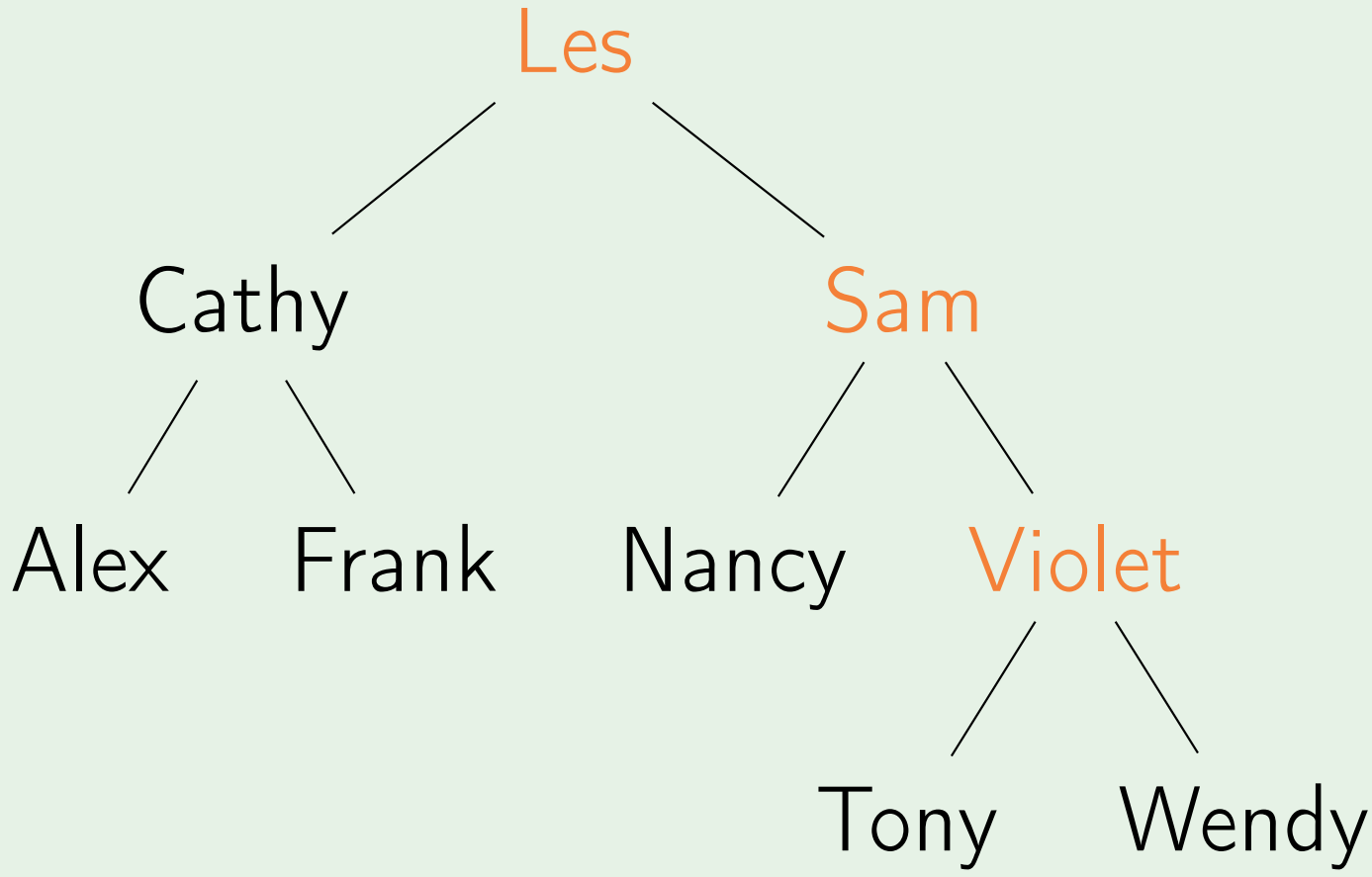
Output: Alex Cathy Frank Les Nancy Sam Tony

InOrderTraversal



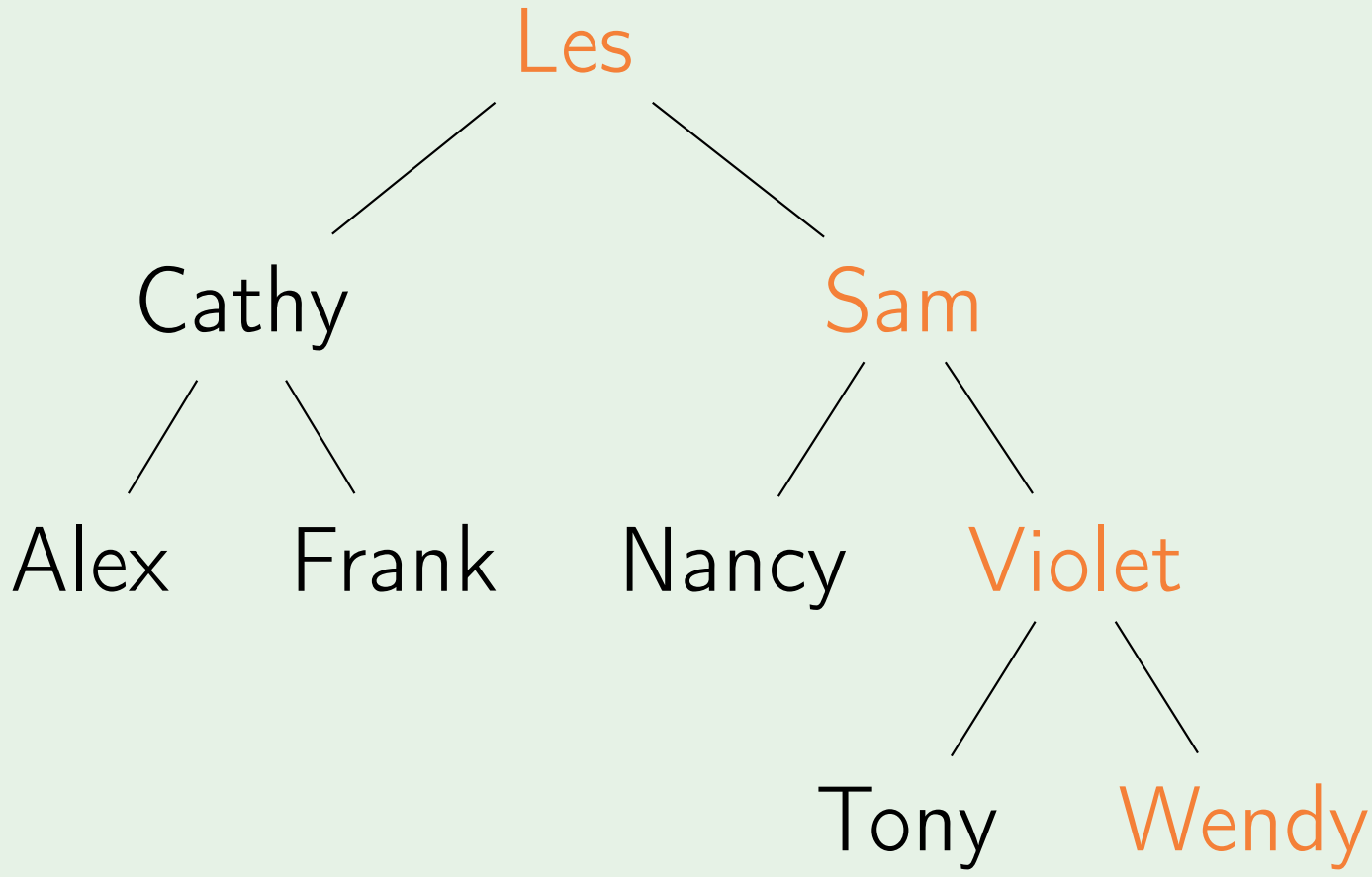
Output: Alex Cathy Frank Les Nancy Sam Tony Violet

InOrderTraversal



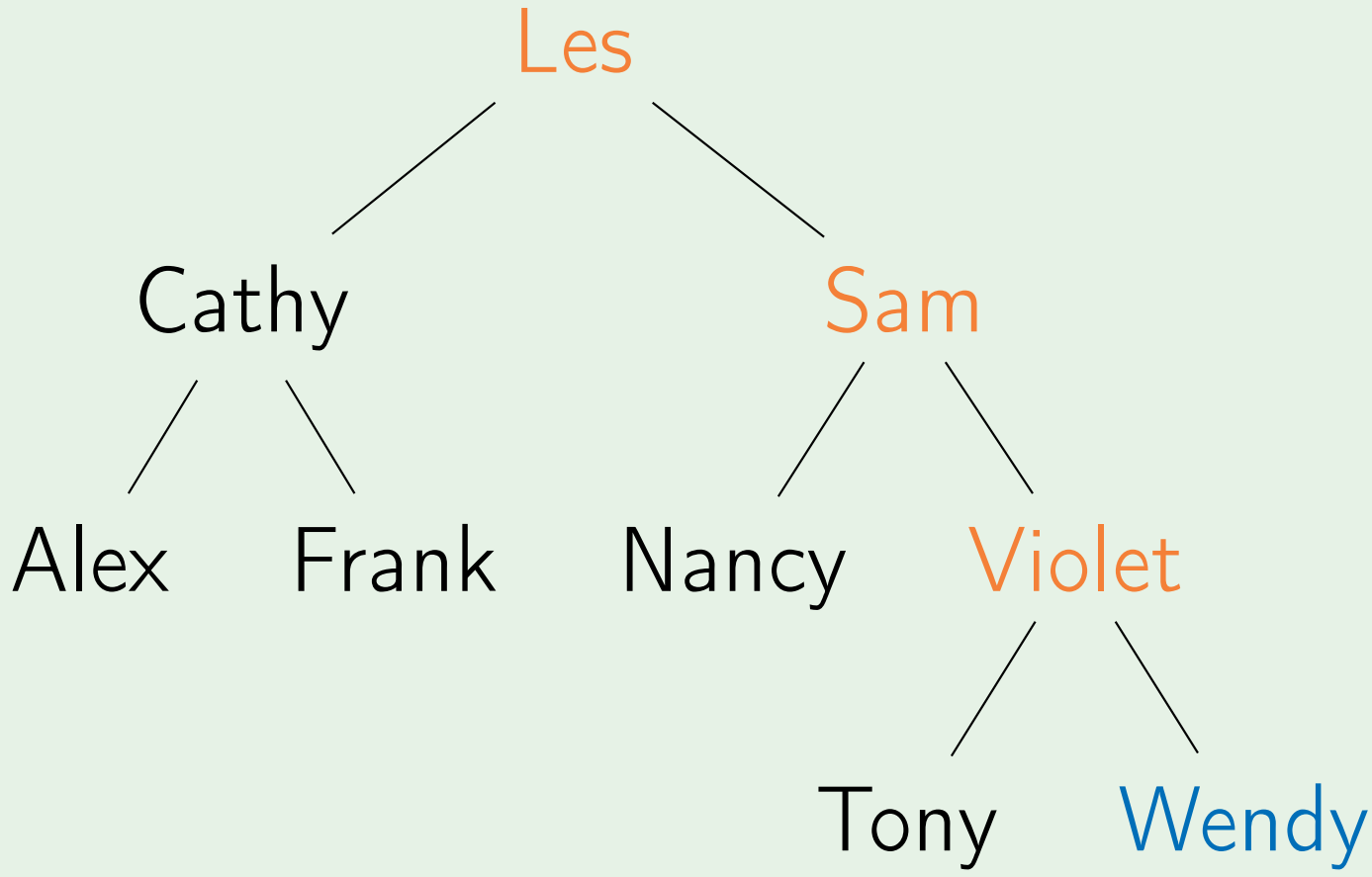
Output: Alex Cathy Frank Les Nancy Sam Tony Violet

InOrderTraversal



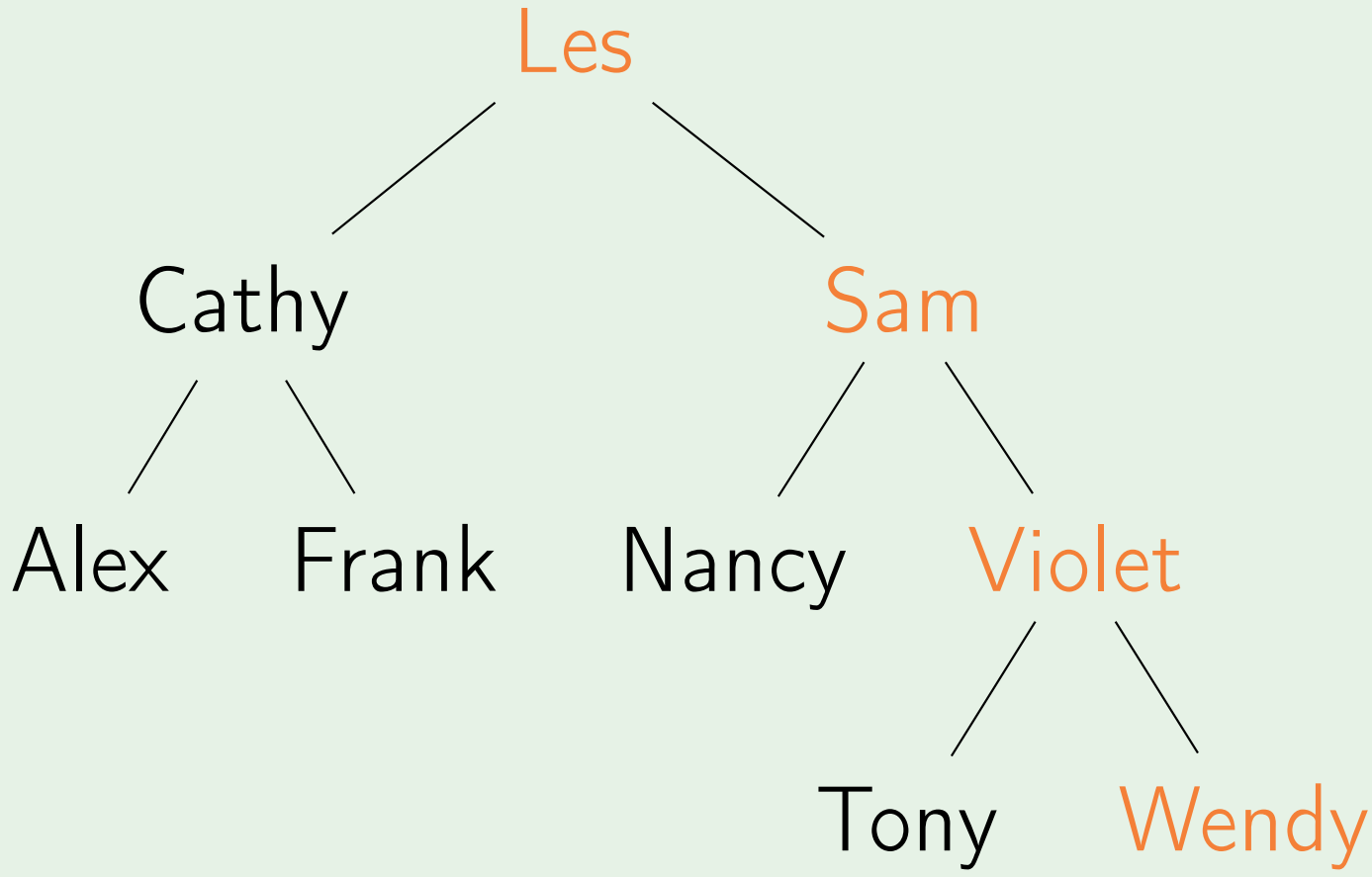
Output: Alex Cathy Frank Les Nancy Sam Tony Violet

InOrderTraversal



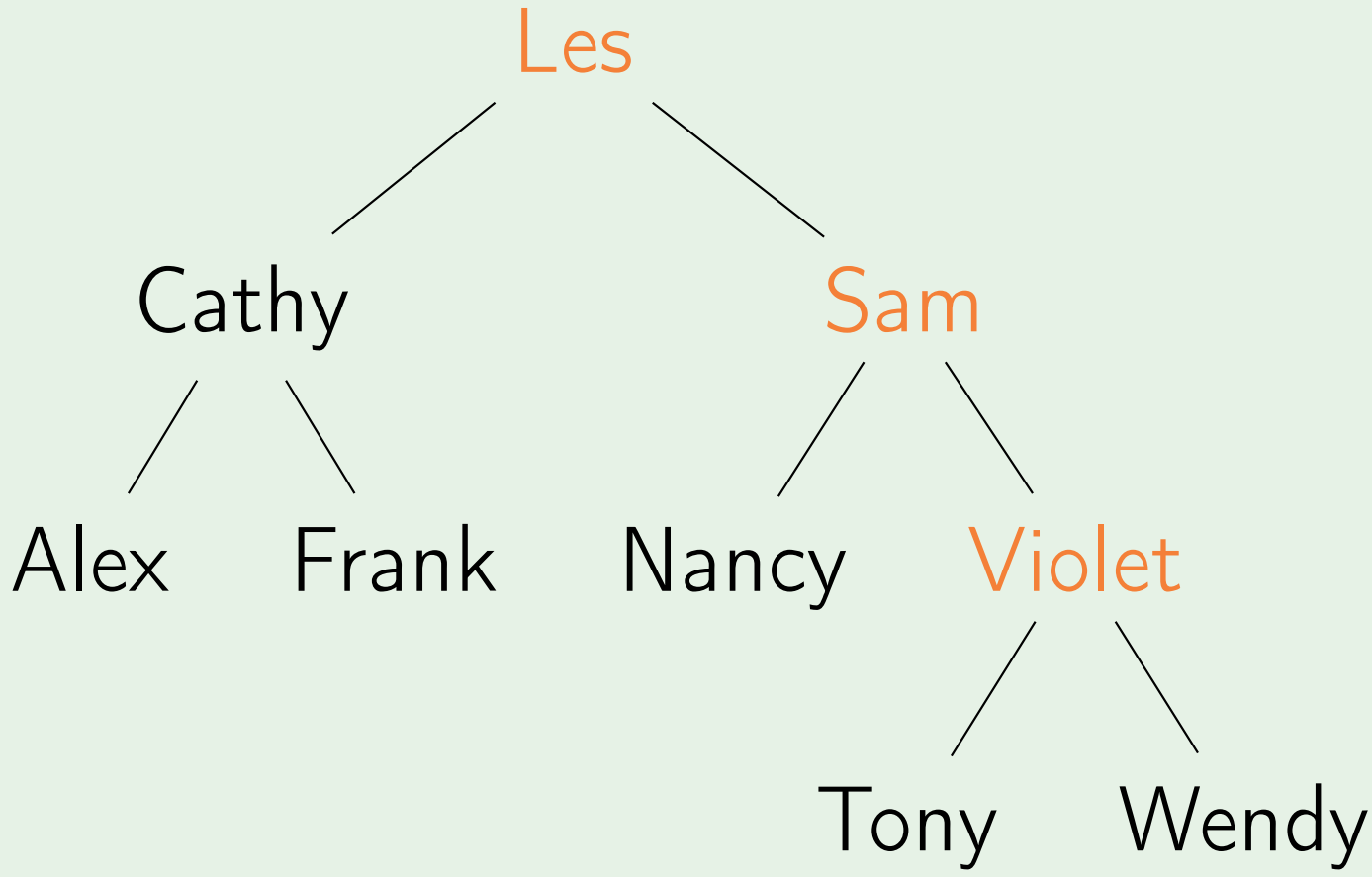
Output: Alex Cathy Frank Les Nancy Sam Tony Violet Wendy

InOrderTraversal



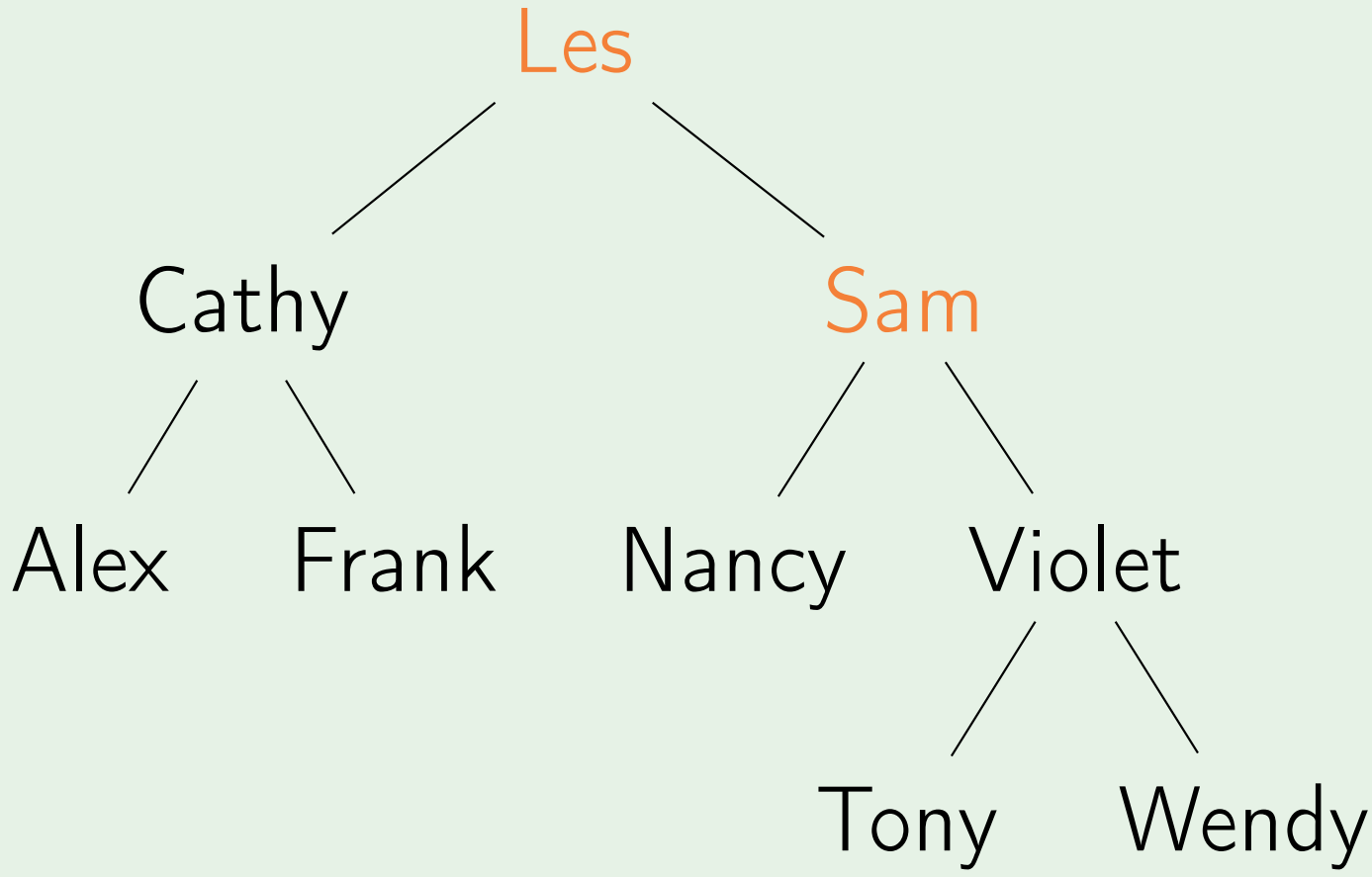
Output: Alex Cathy Frank Les Nancy Sam Tony Violet Wendy

InOrderTraversal



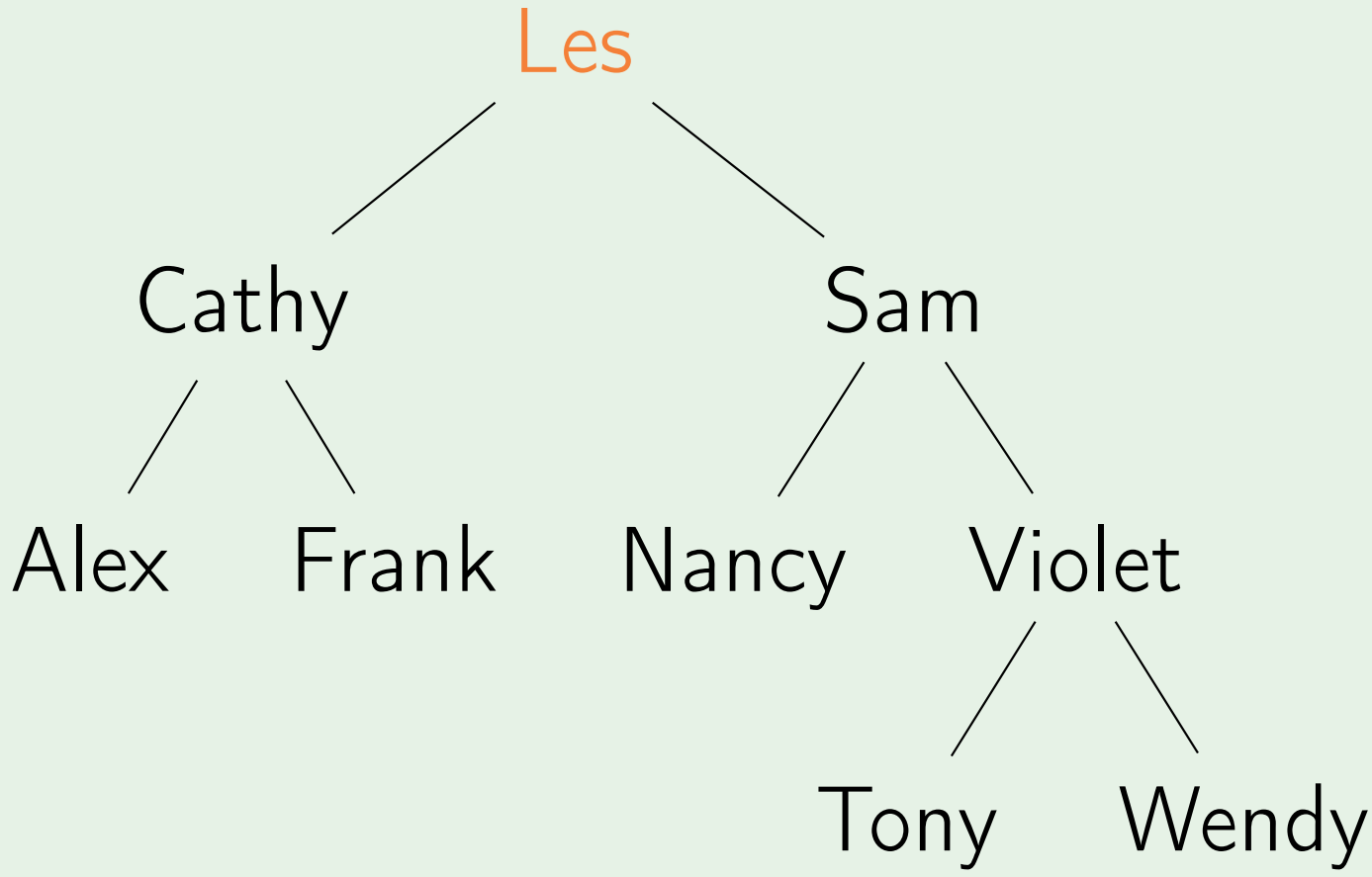
Output: Alex Cathy Frank Les Nancy Sam Tony Violet Wendy

InOrderTraversal



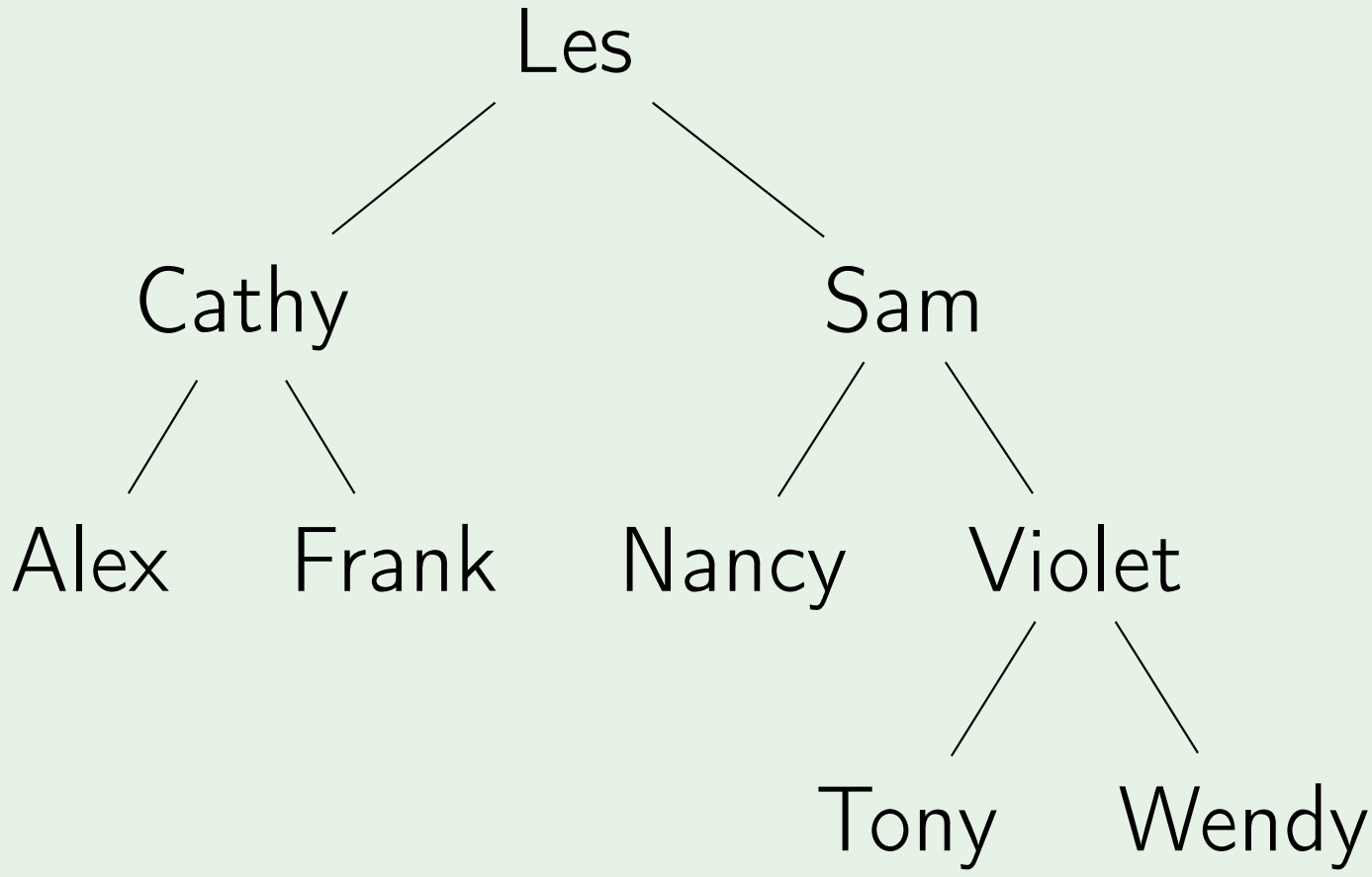
Output: Alex Cathy Frank Les Nancy Sam Tony Violet Wendy

InOrderTraversal



Output: Alex Cathy Frank Les Nancy Sam Tony Violet Wendy

InOrderTraversal

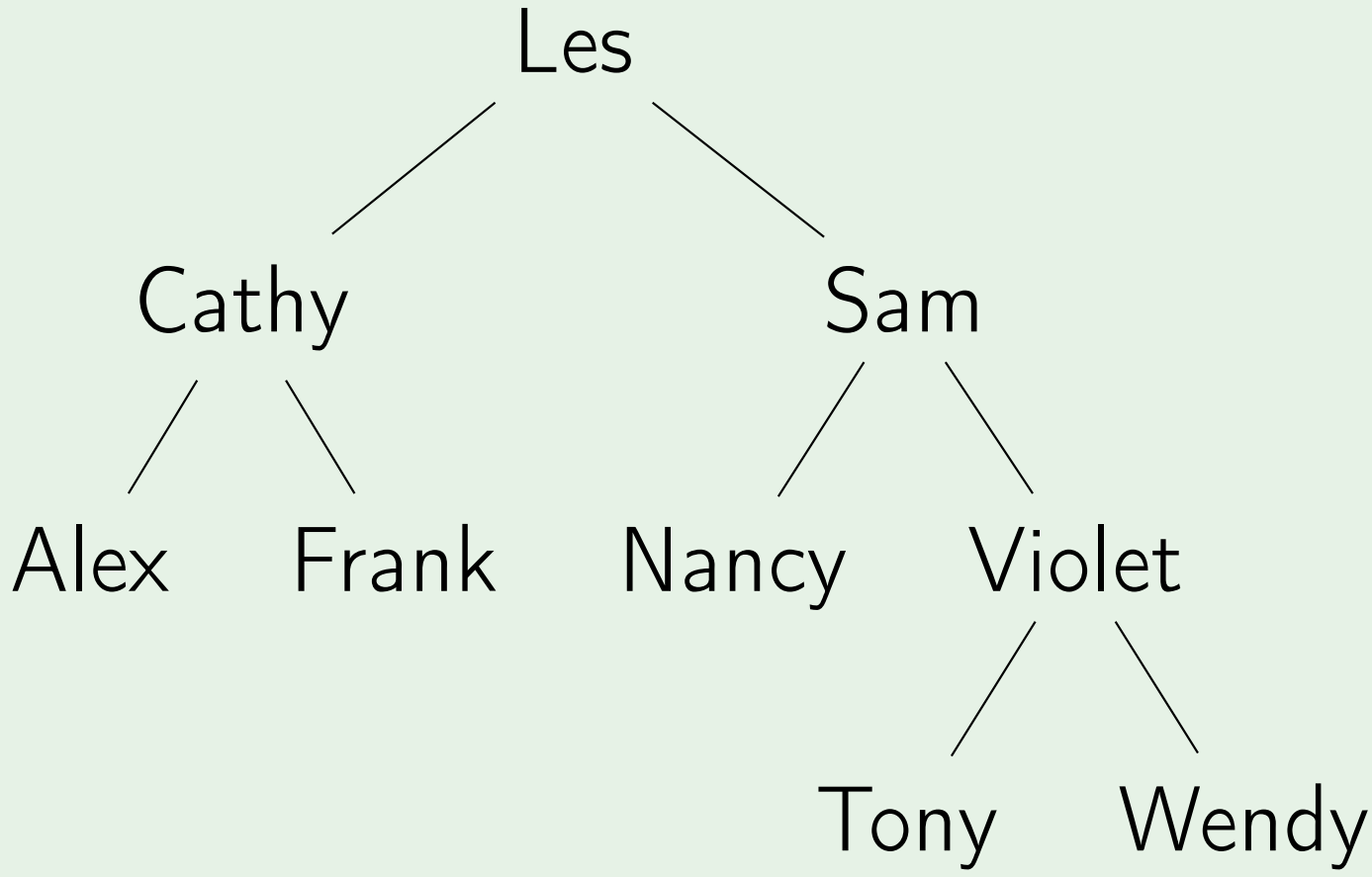


Output: Alex Cathy Frank Les Nancy Sam Tony Violet Wendy

Depth-first

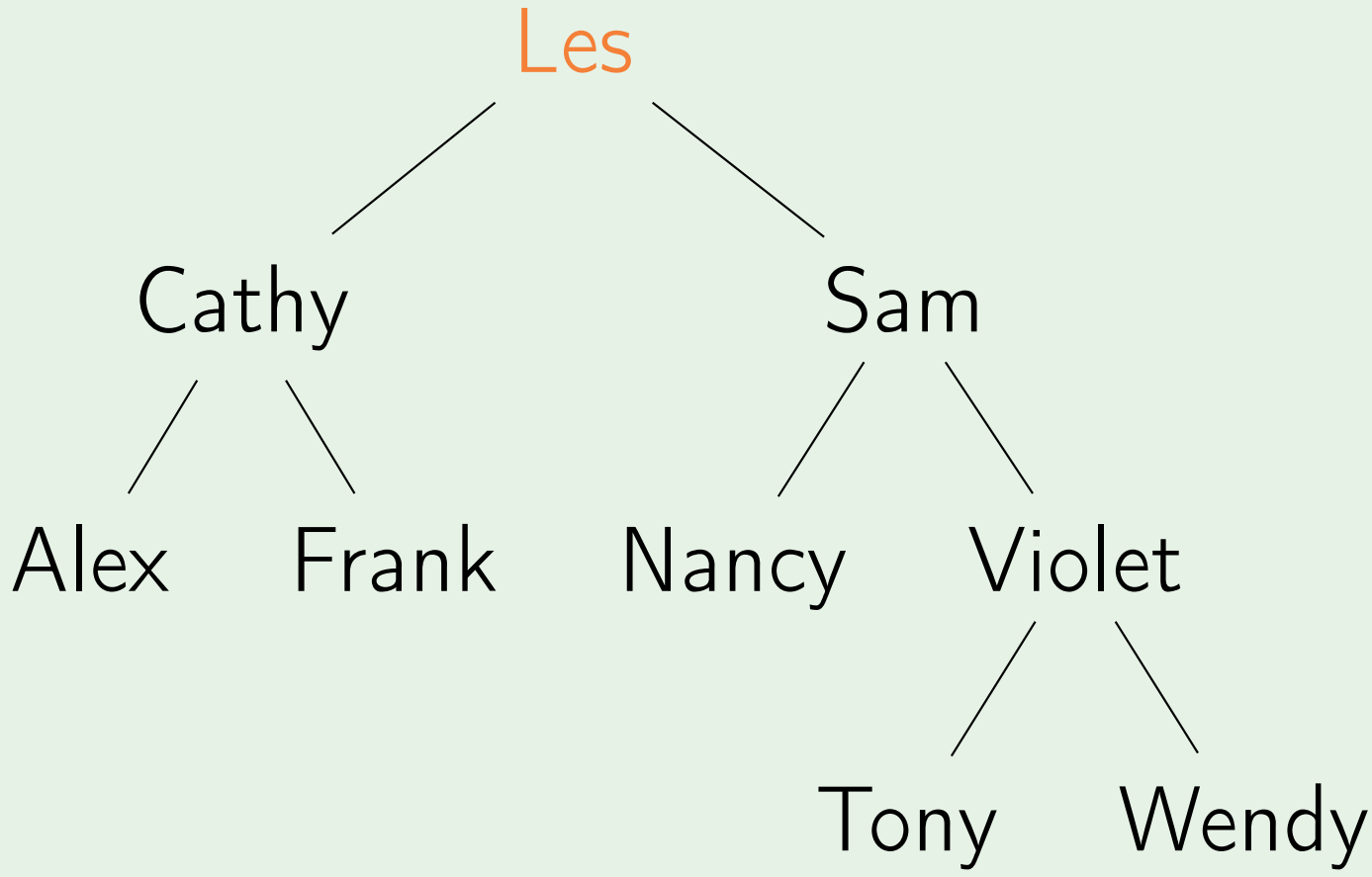
```
PreOrderTraversal(tree)
if tree = nil :
   return
Print(tree.key)
PreOrderTraversal(tree.left)
PreOrderTraversal(tree.right)
```

PreOrderTraversal



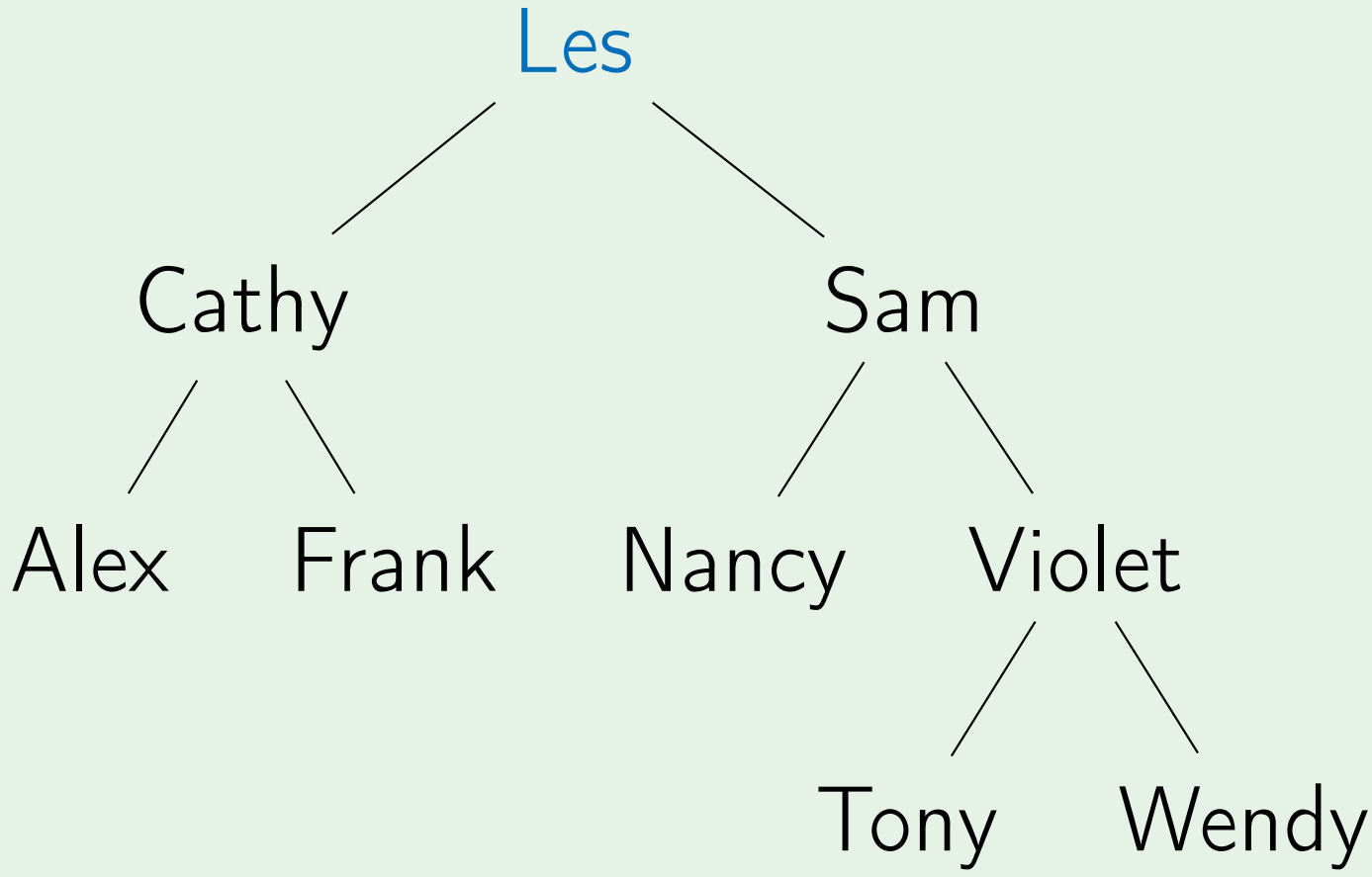
Output:

PreOrderTraversal



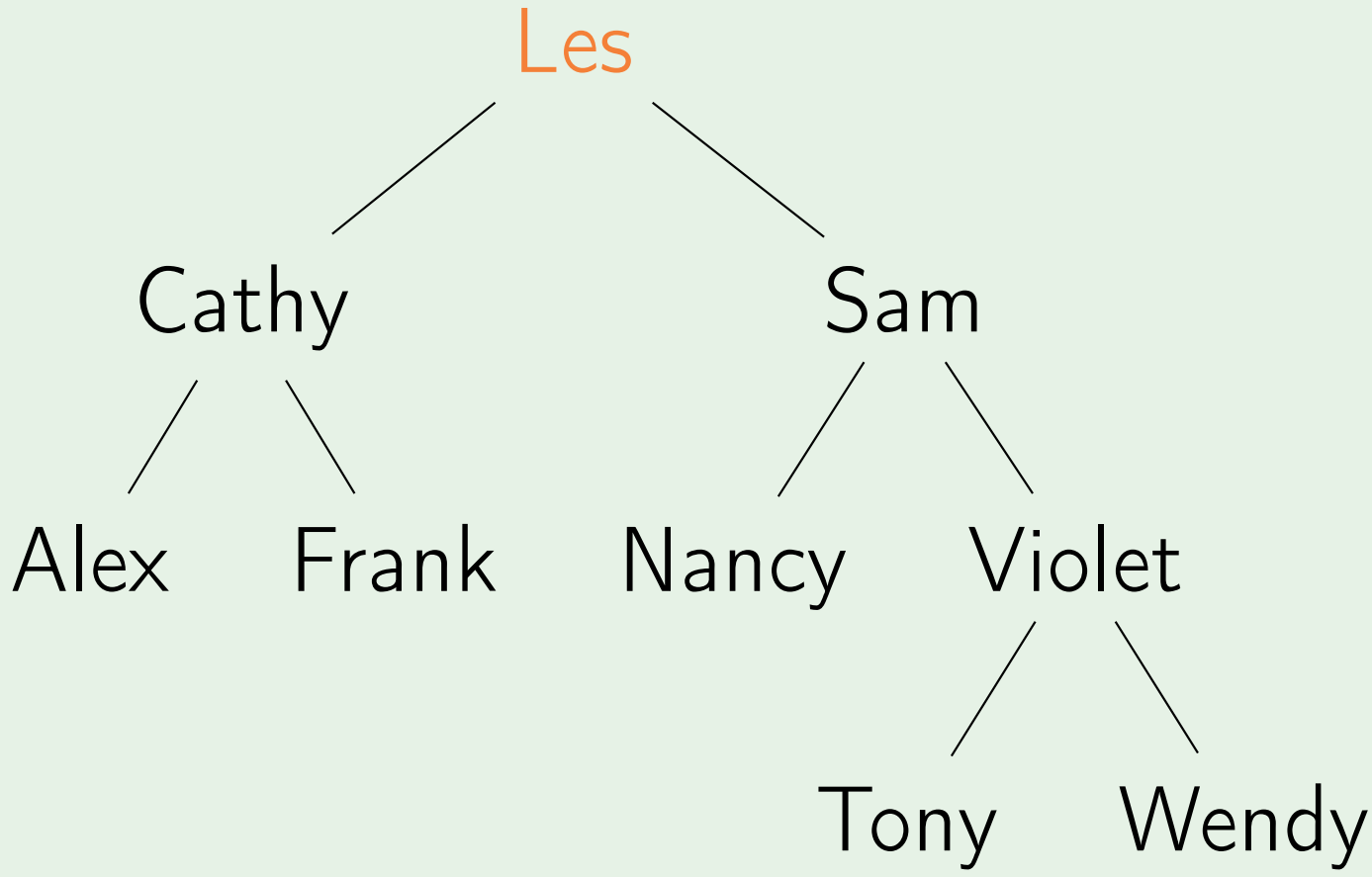
Output:

PreOrderTraversal



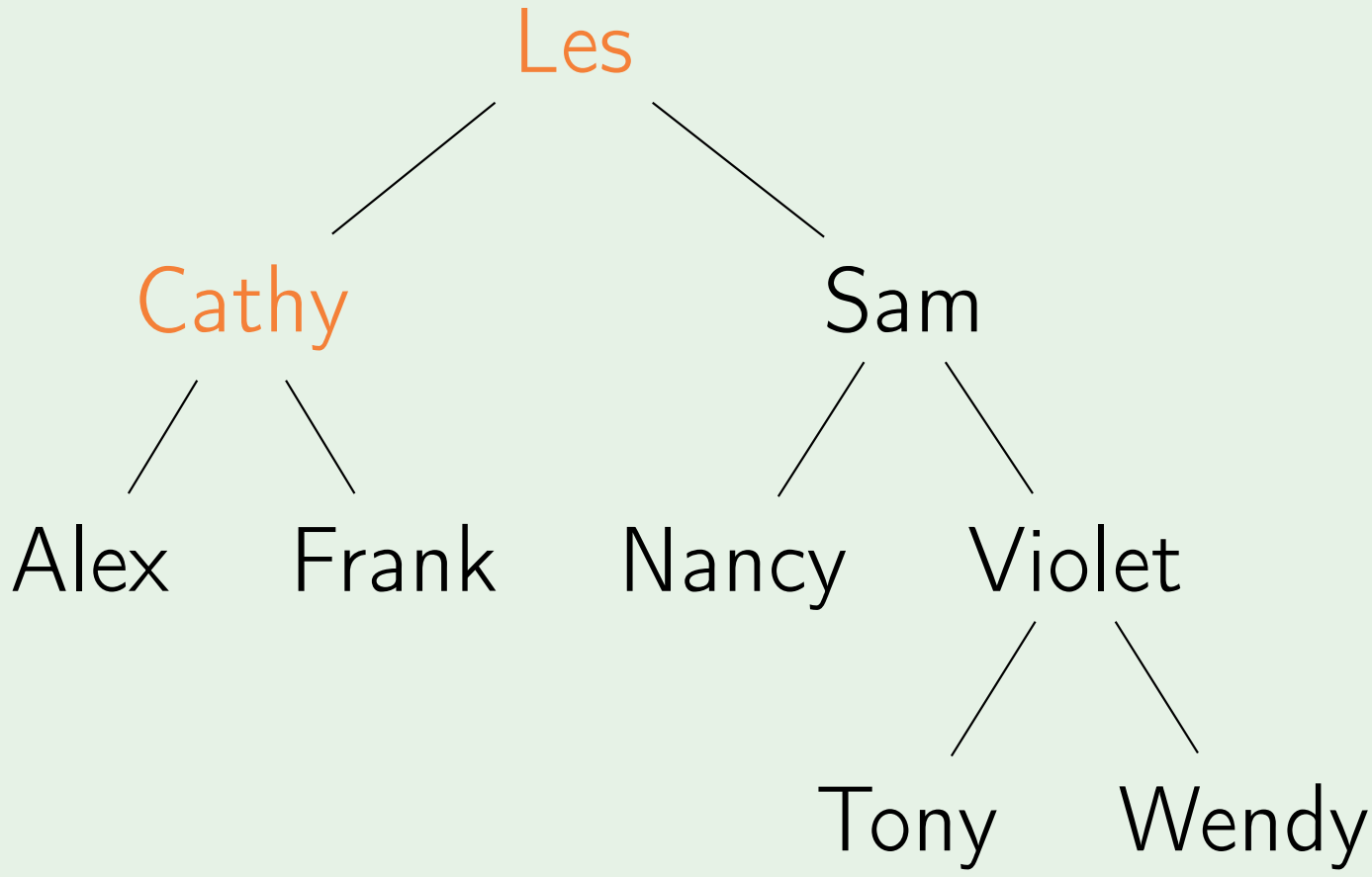
Output: Les

PreOrderTraversal



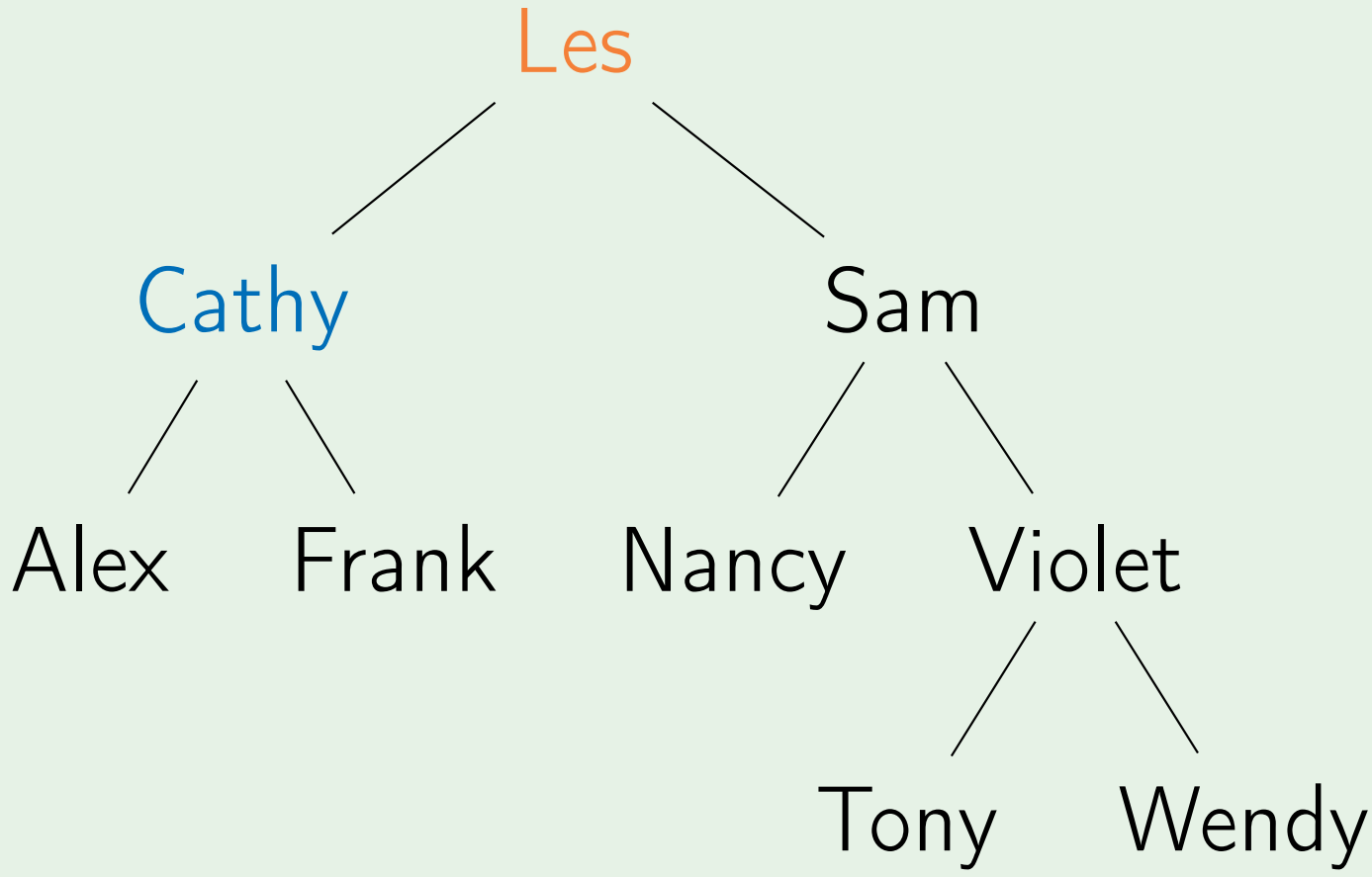
Output: Les

PreOrderTraversal



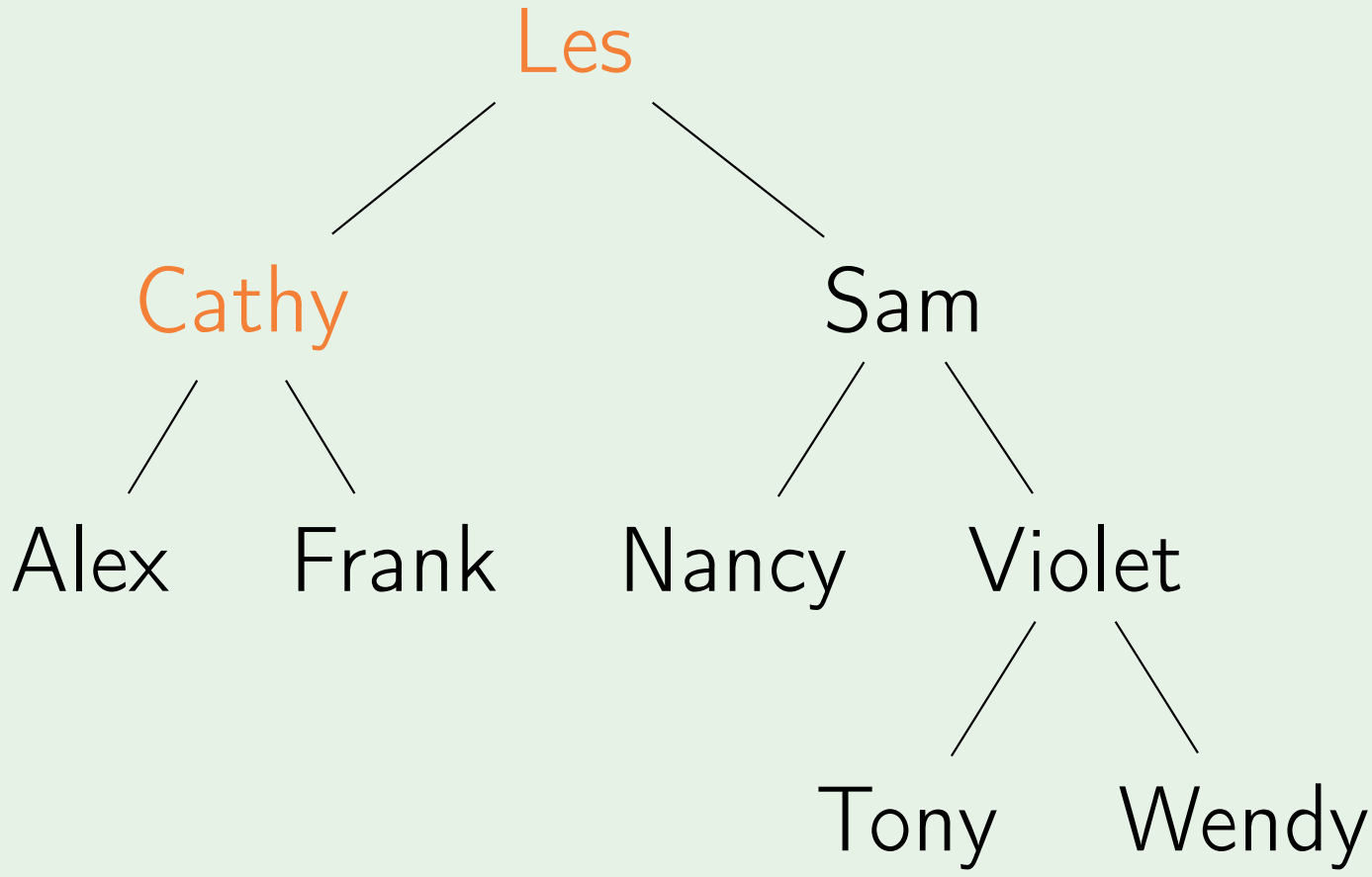
Output: Les

PreOrderTraversal



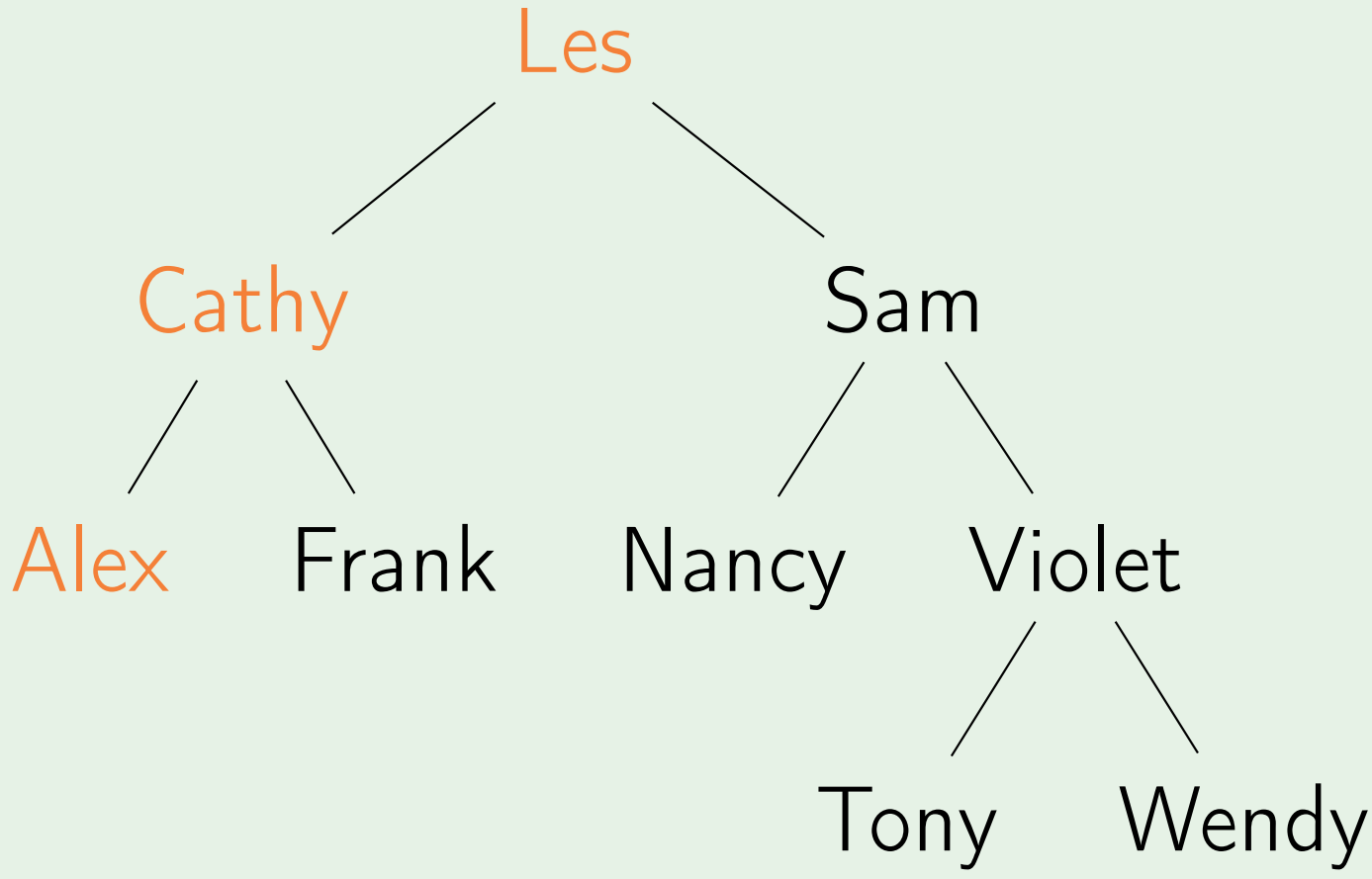
Output: Les Cathy

PreOrderTraversal



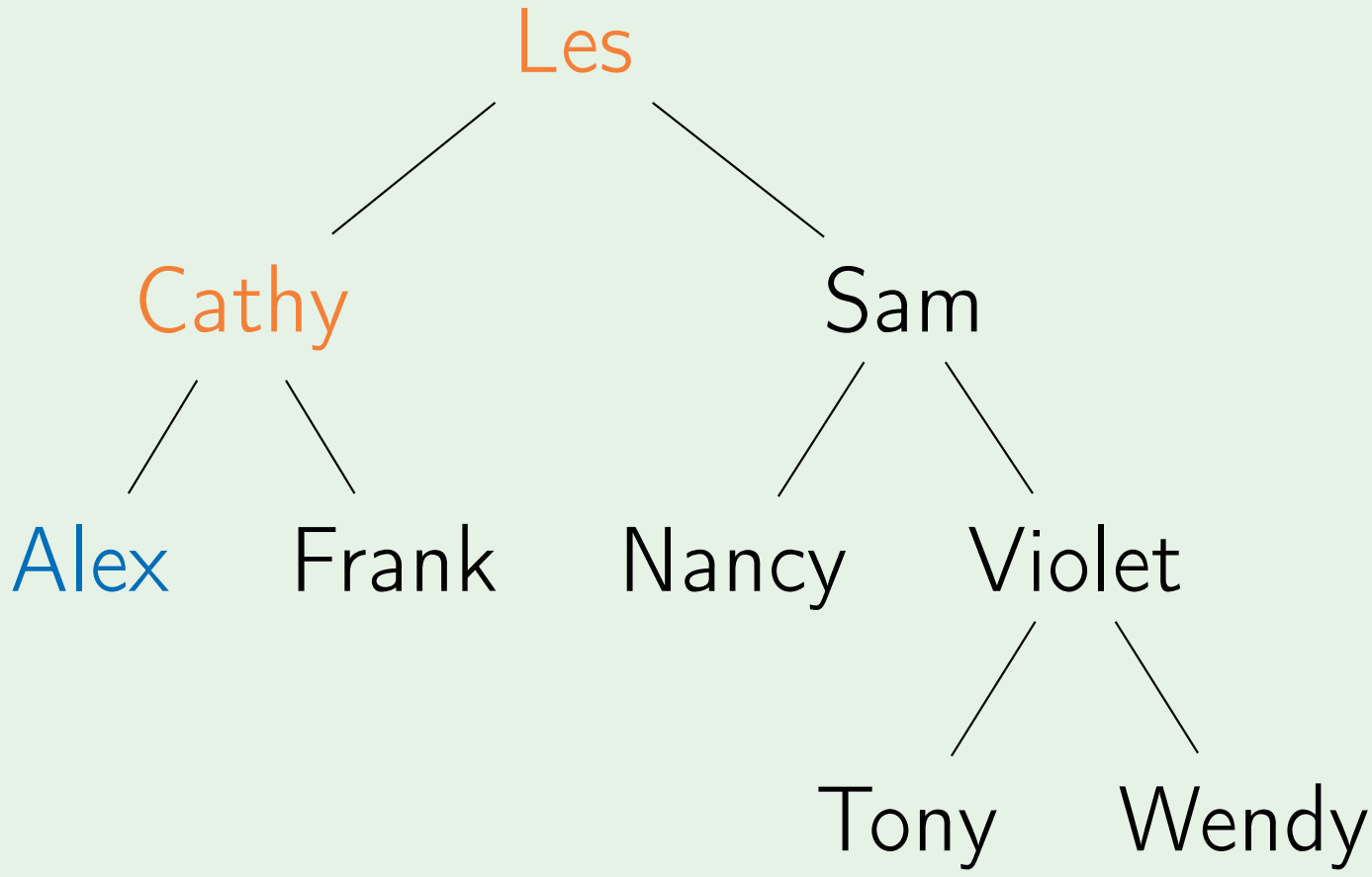
Output: Les Cathy

PreOrderTraversal



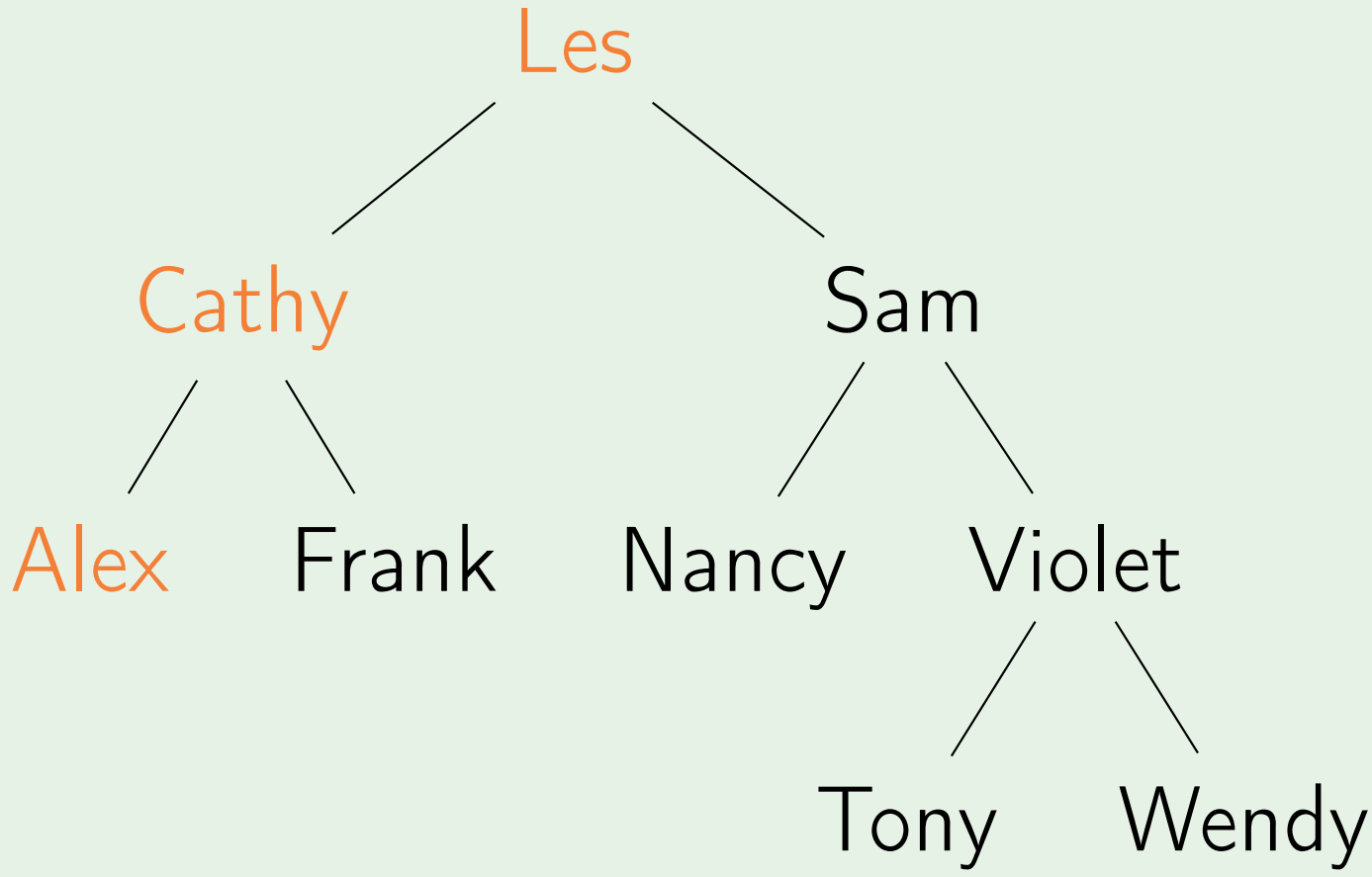
Output: Les Cathy

PreOrderTraversal



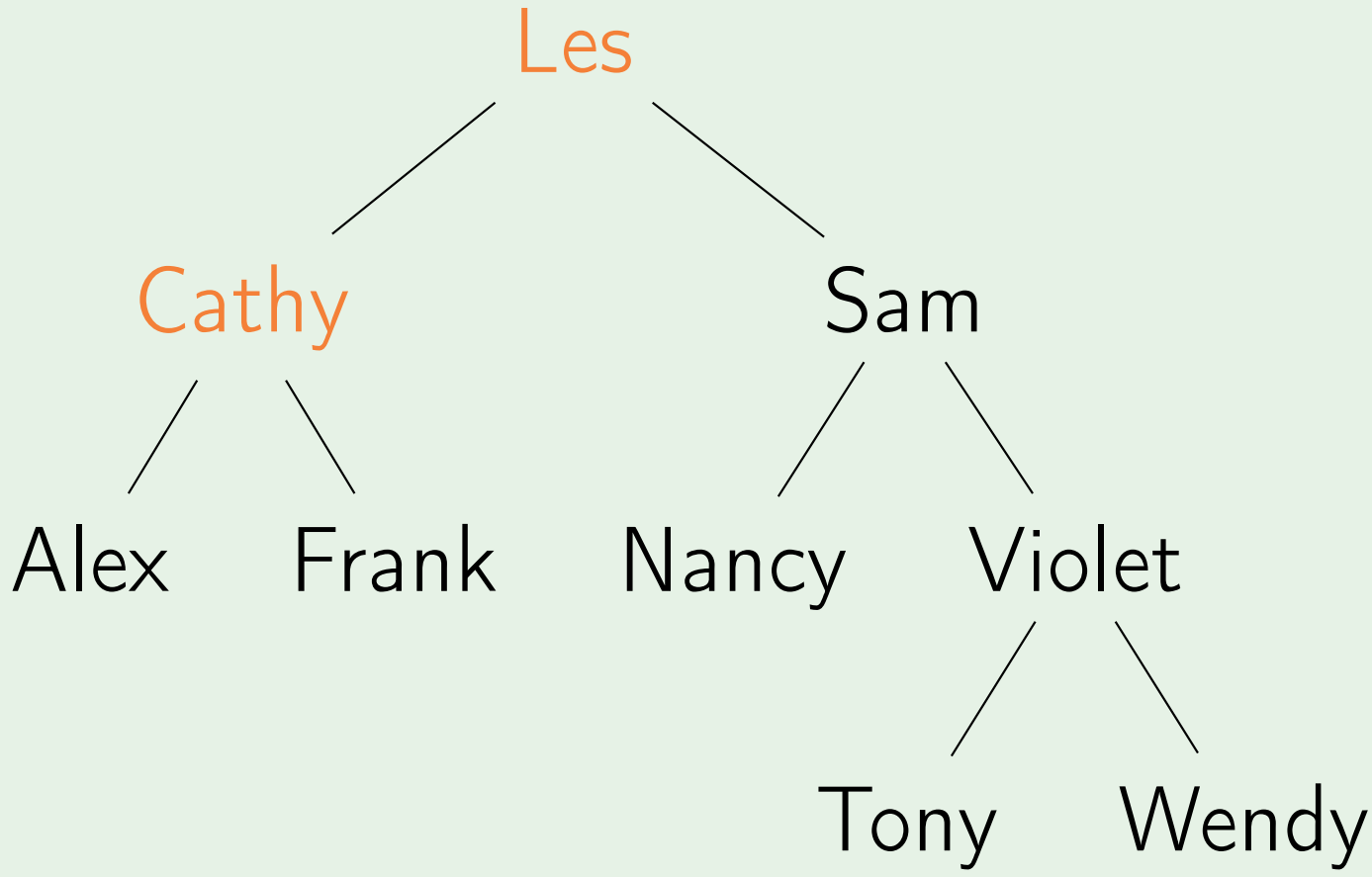
Output: Les Cathy Alex

PreOrderTraversal



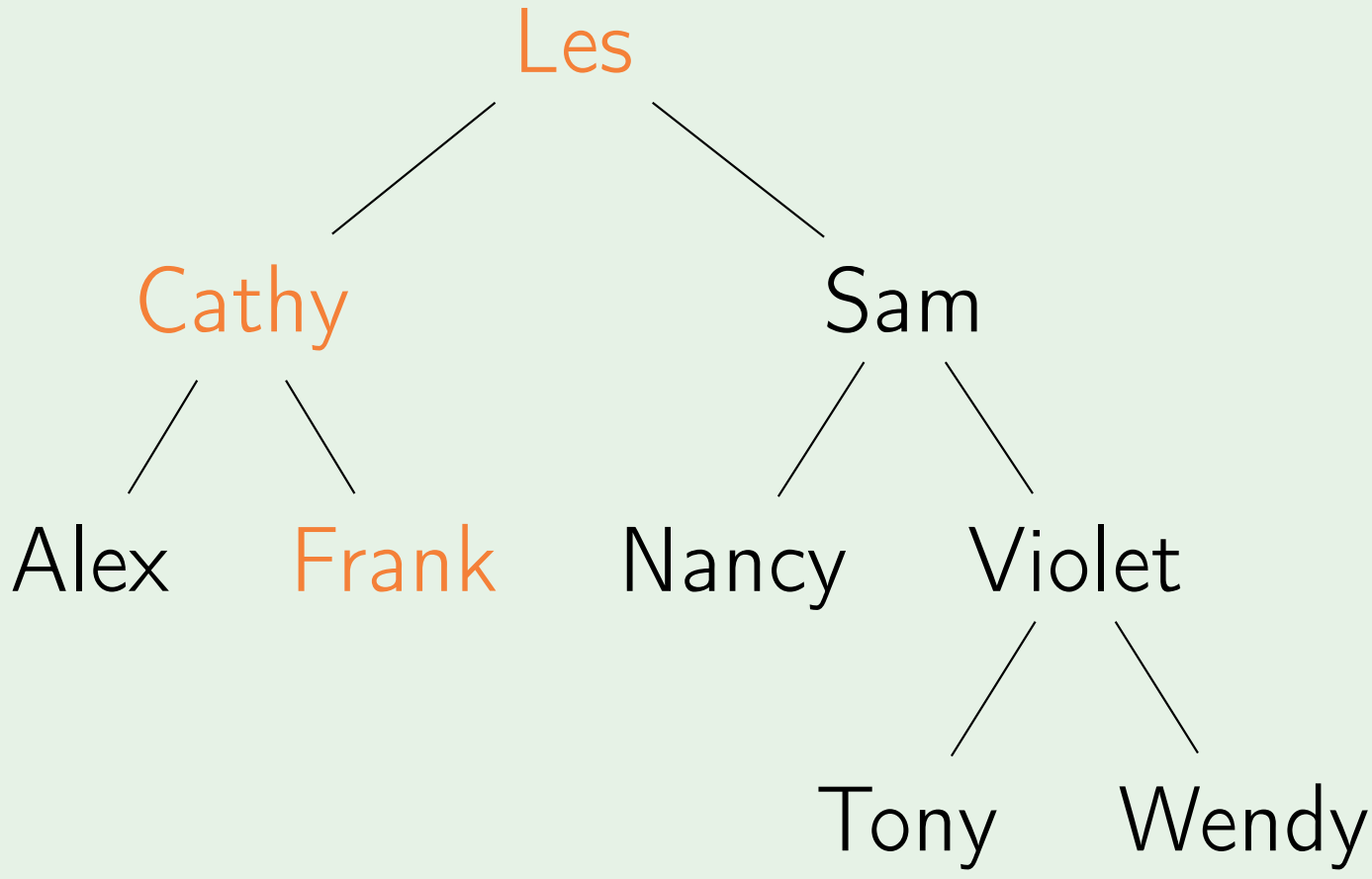
Output: Les Cathy Alex

PreOrderTraversal



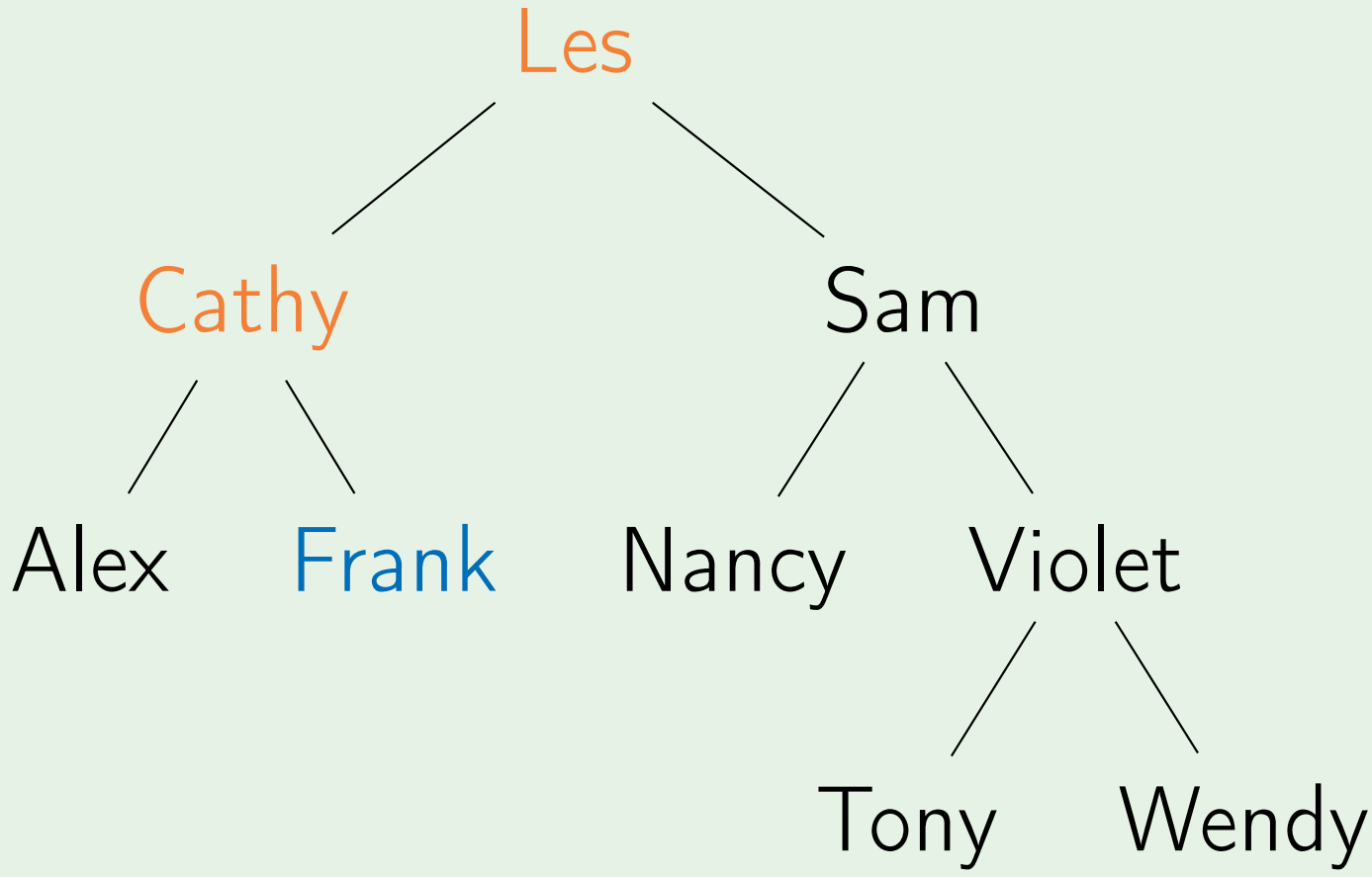
Output: Les Cathy Alex

PreOrderTraversal



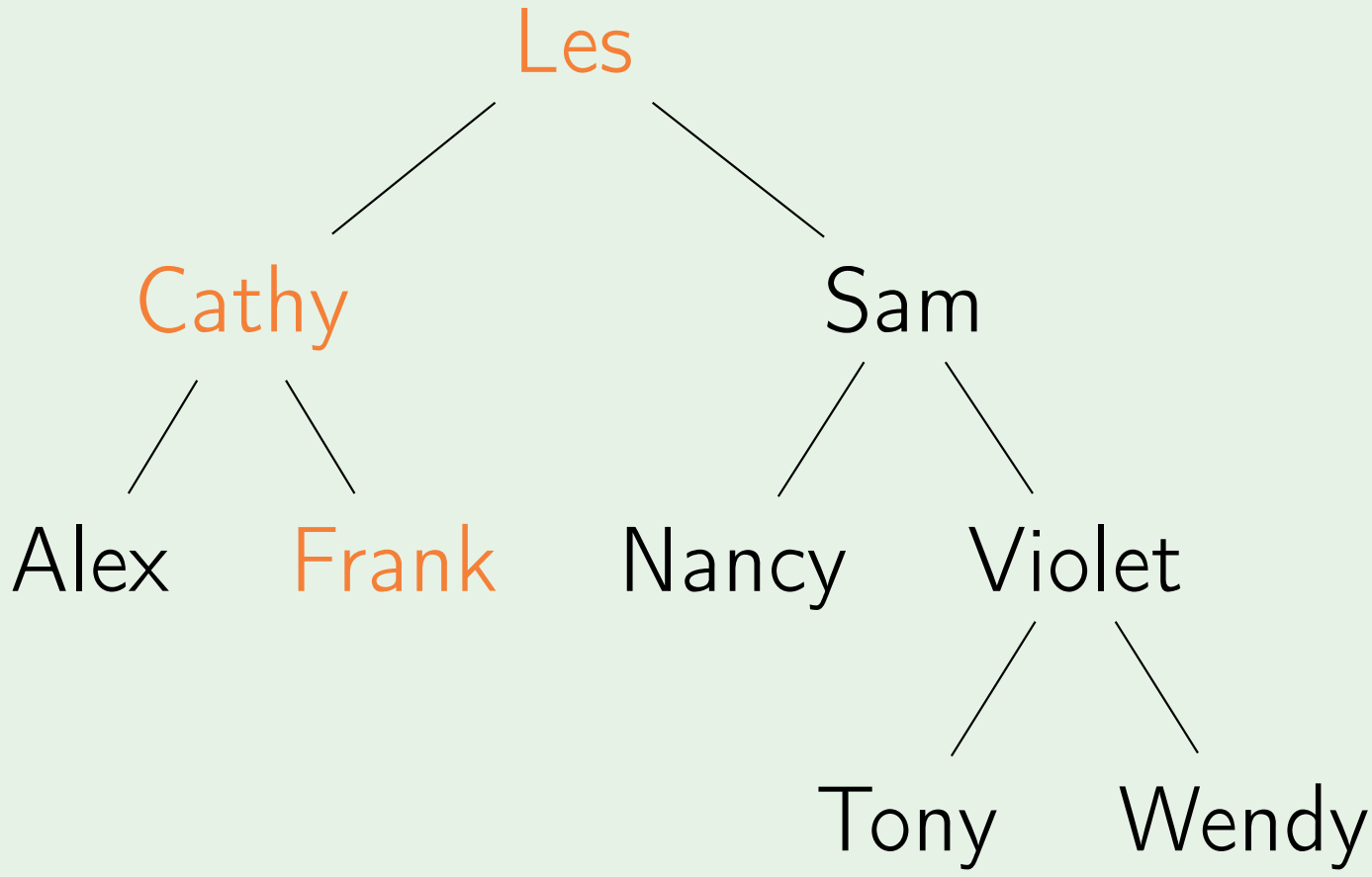
Output: Les Cathy Alex

PreOrderTraversal



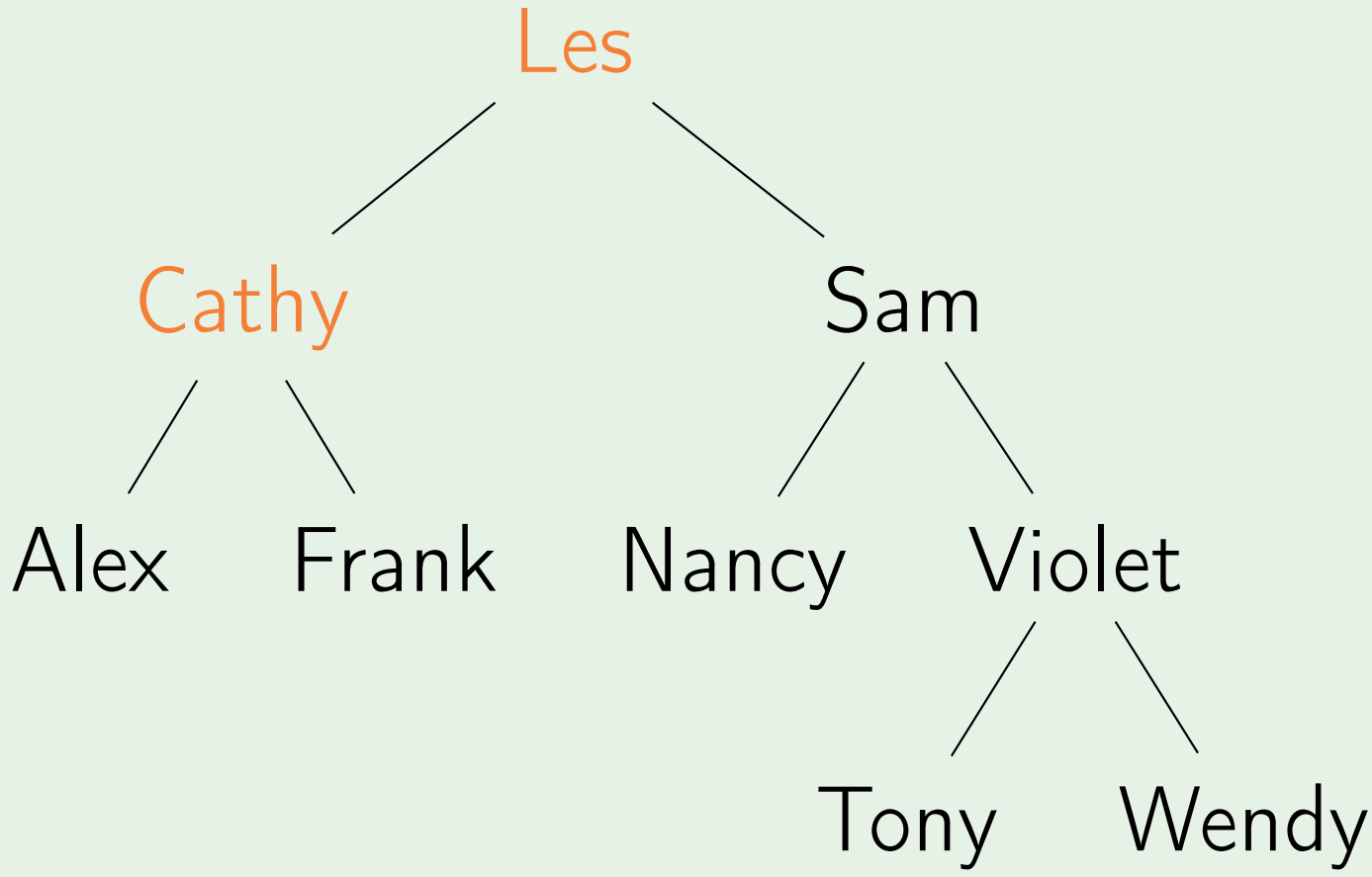
Output: Les Cathy Alex Frank

PreOrderTraversal



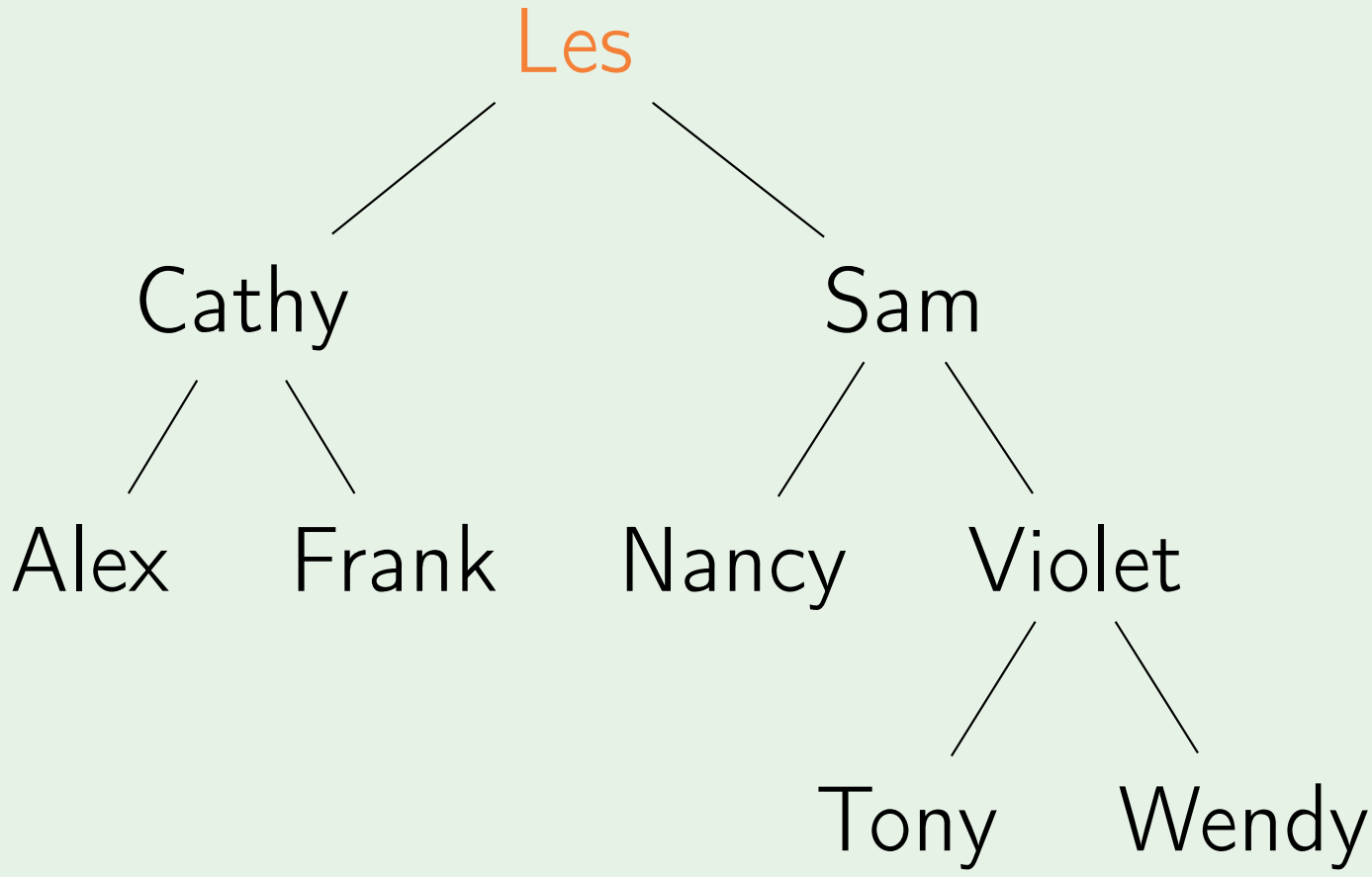
Output: Les Cathy Alex Frank

PreOrderTraversal



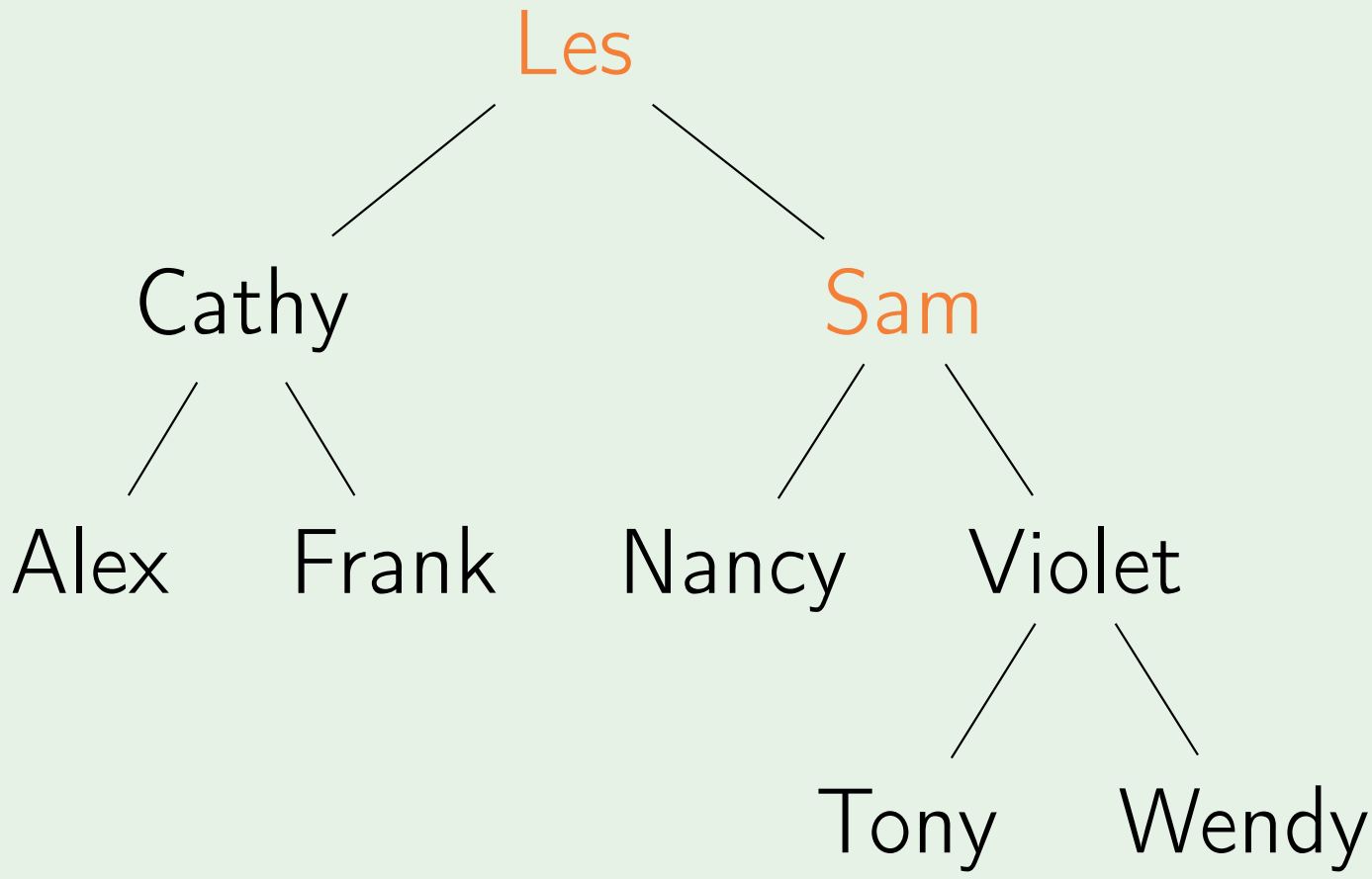
Output: Les Cathy Alex Frank

PreOrderTraversal



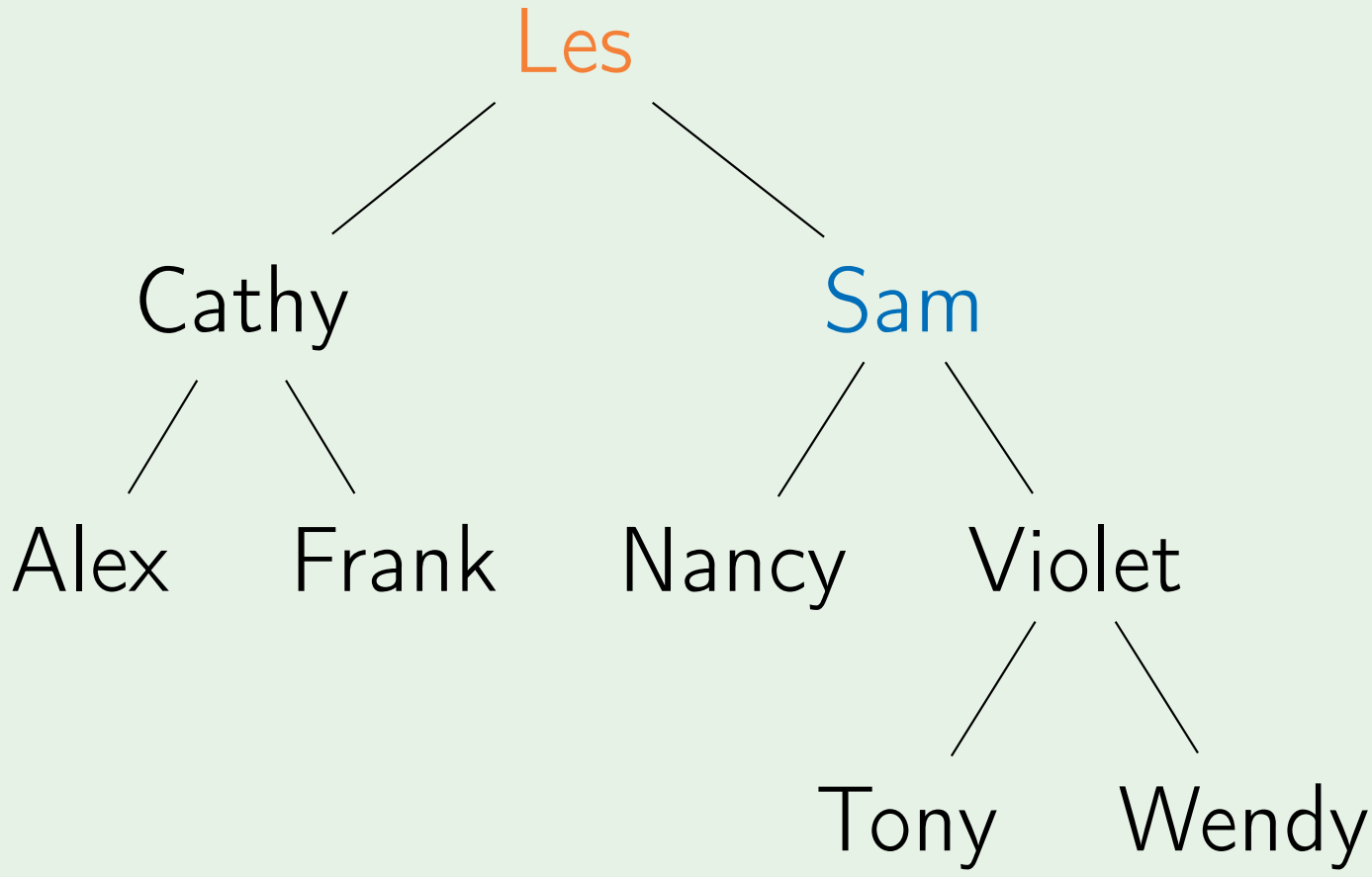
Output: Les Cathy Alex Frank

PreOrderTraversal



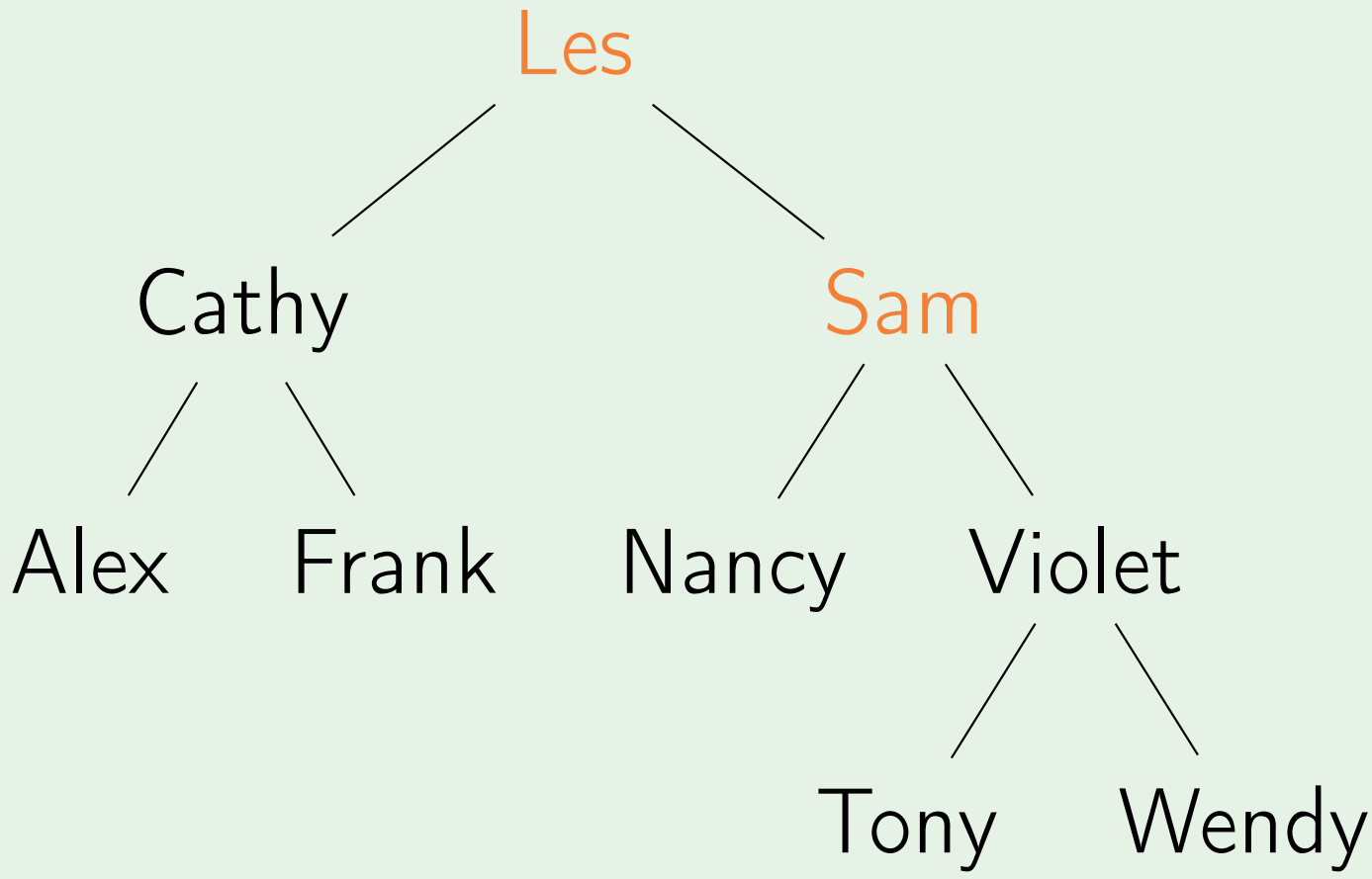
Output: Les Cathy Alex Frank

PreOrderTraversal



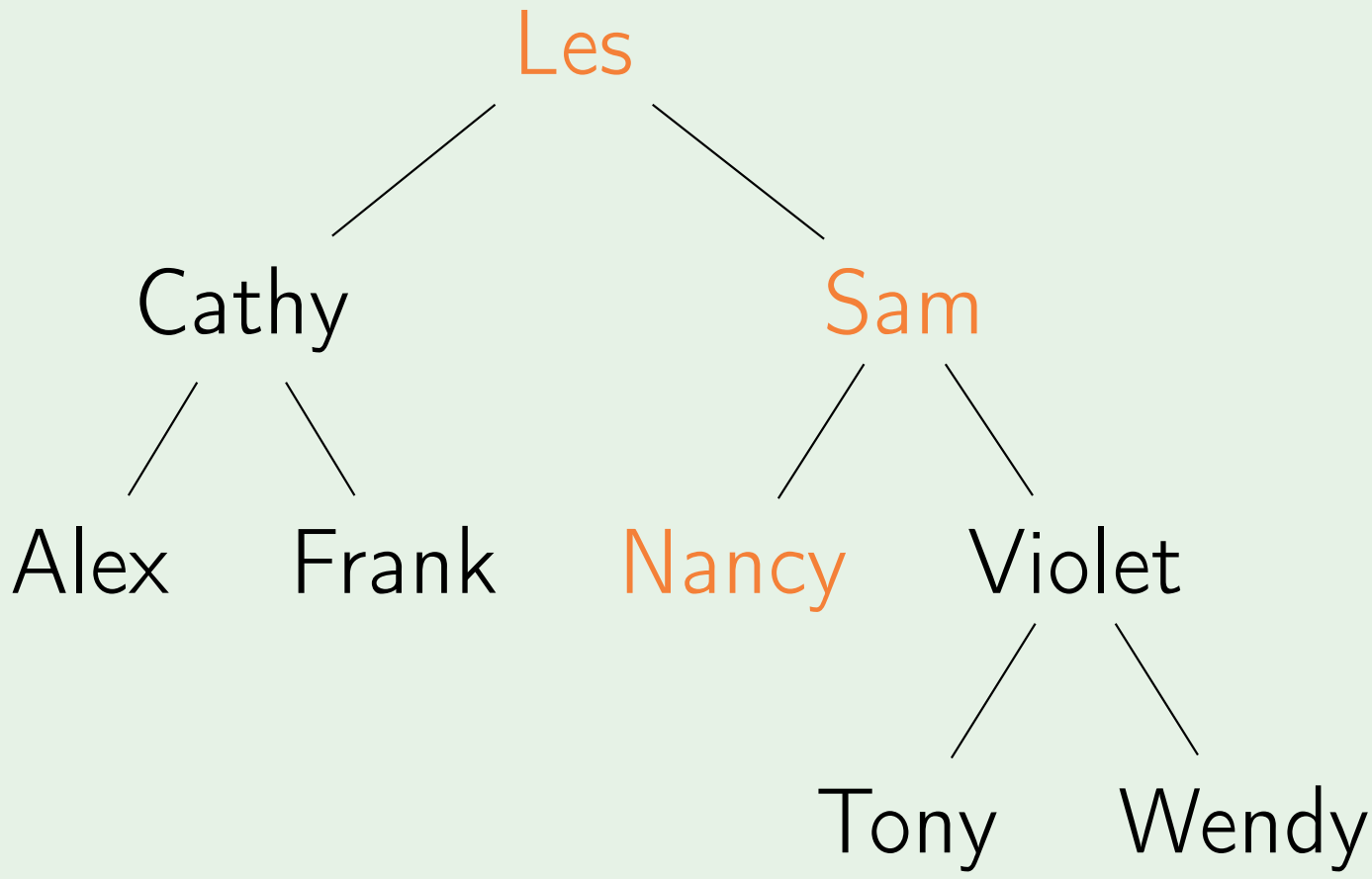
Output: Les Cathy Alex Frank Sam

PreOrderTraversal



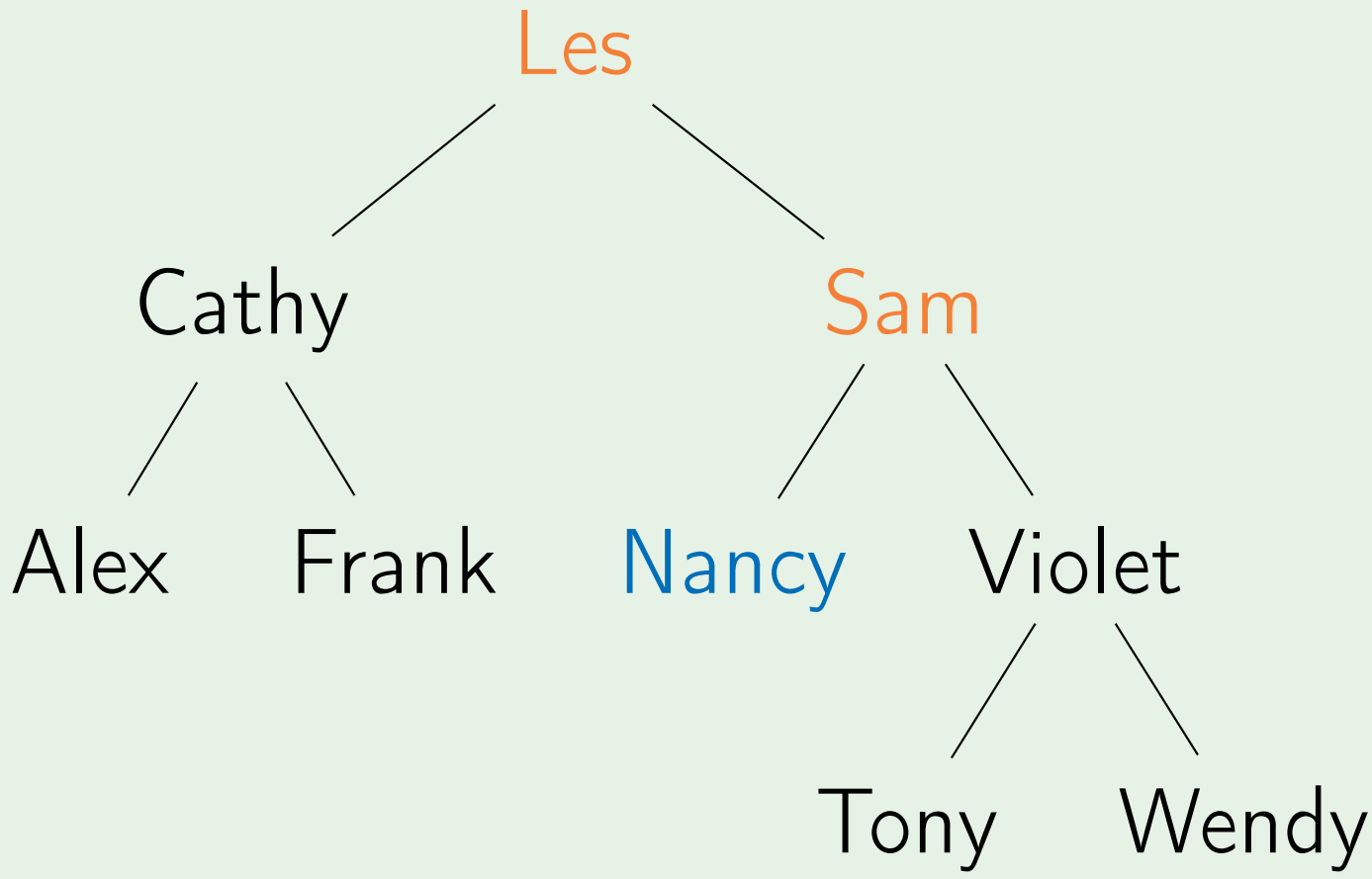
Output: Les Cathy Alex Frank Sam

PreOrderTraversal



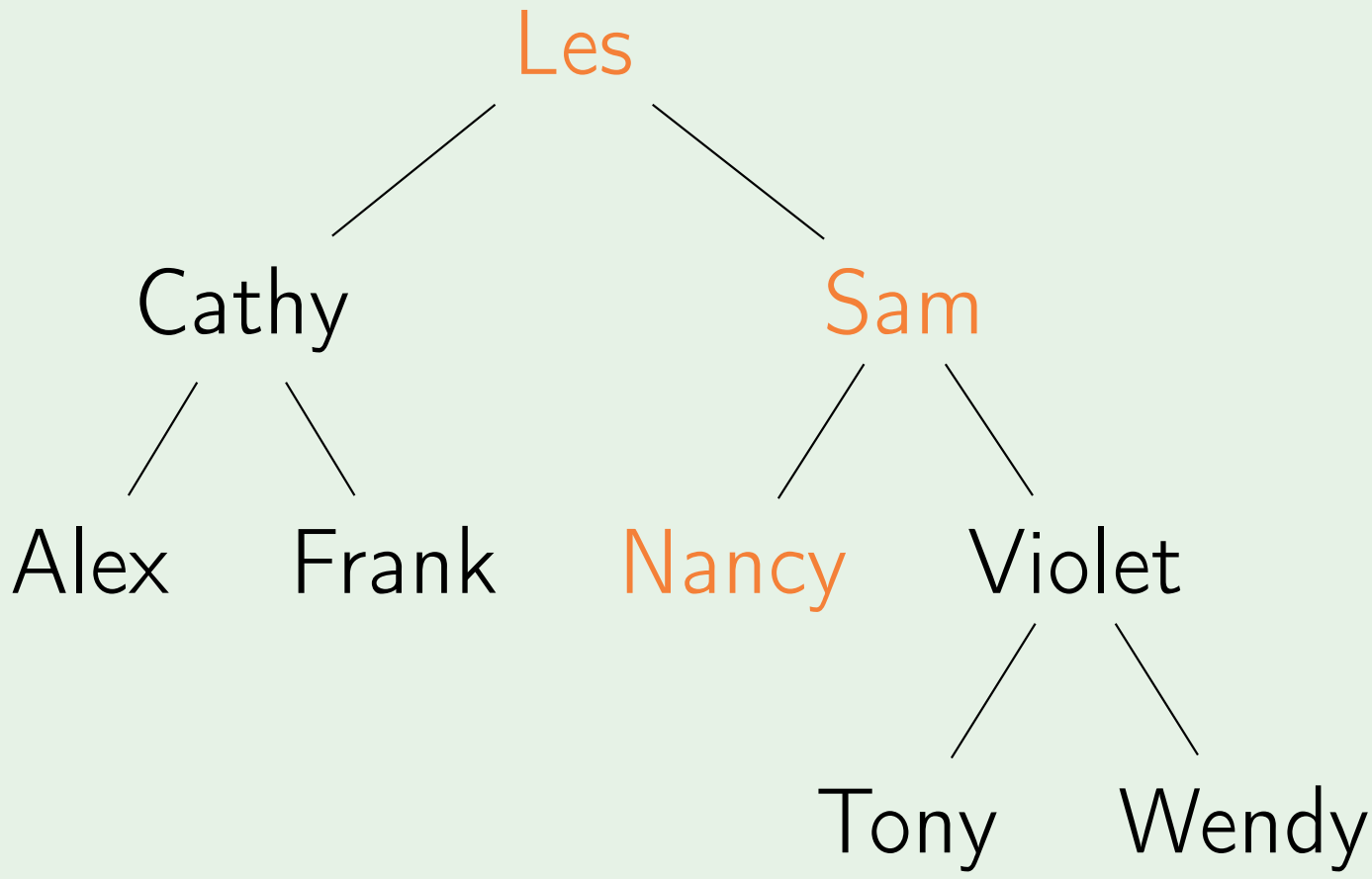
Output: Les Cathy Alex Frank Sam

PreOrderTraversal



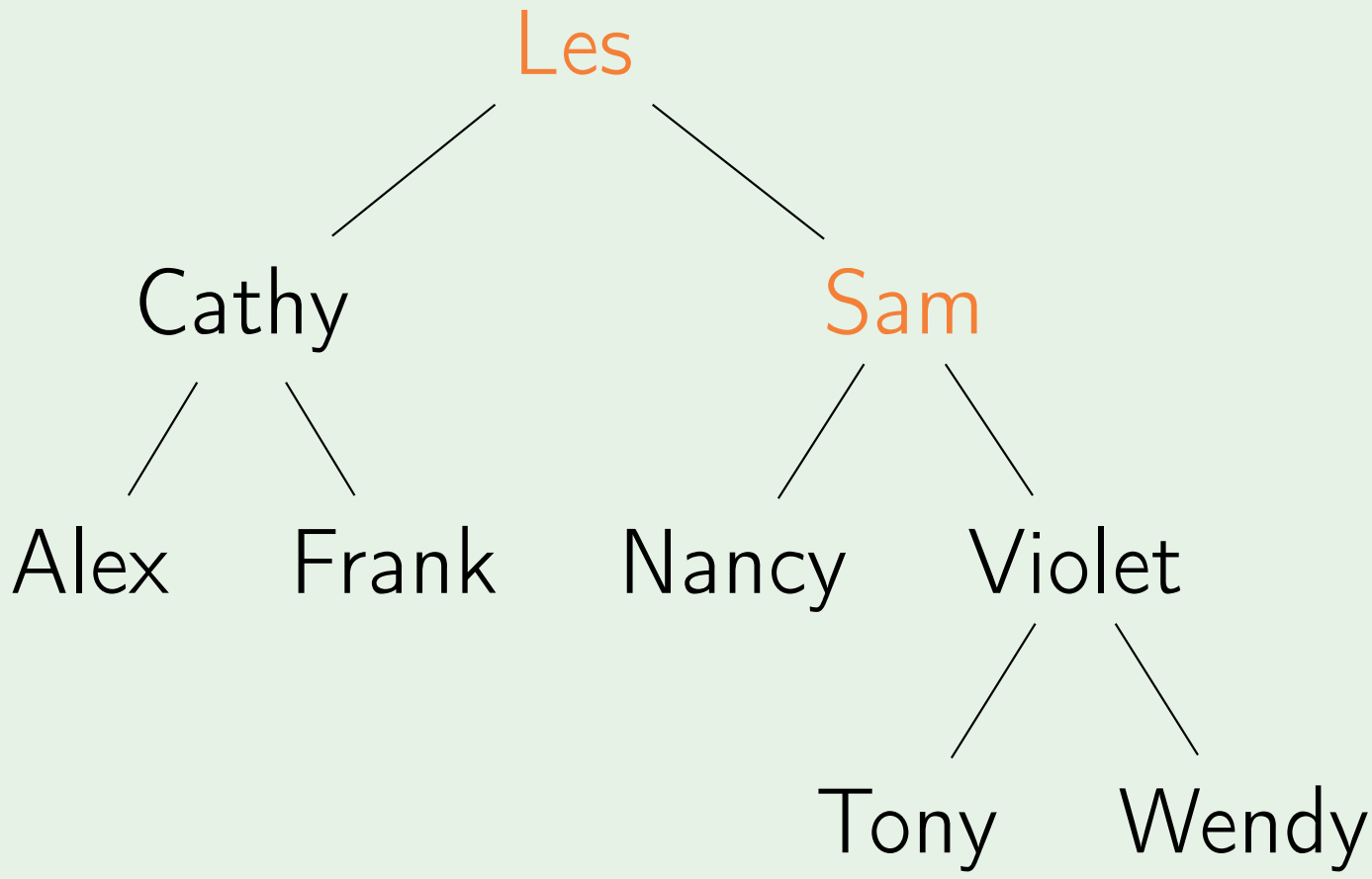
Output: Les Cathy Alex Frank Sam Nancy

PreOrderTraversal



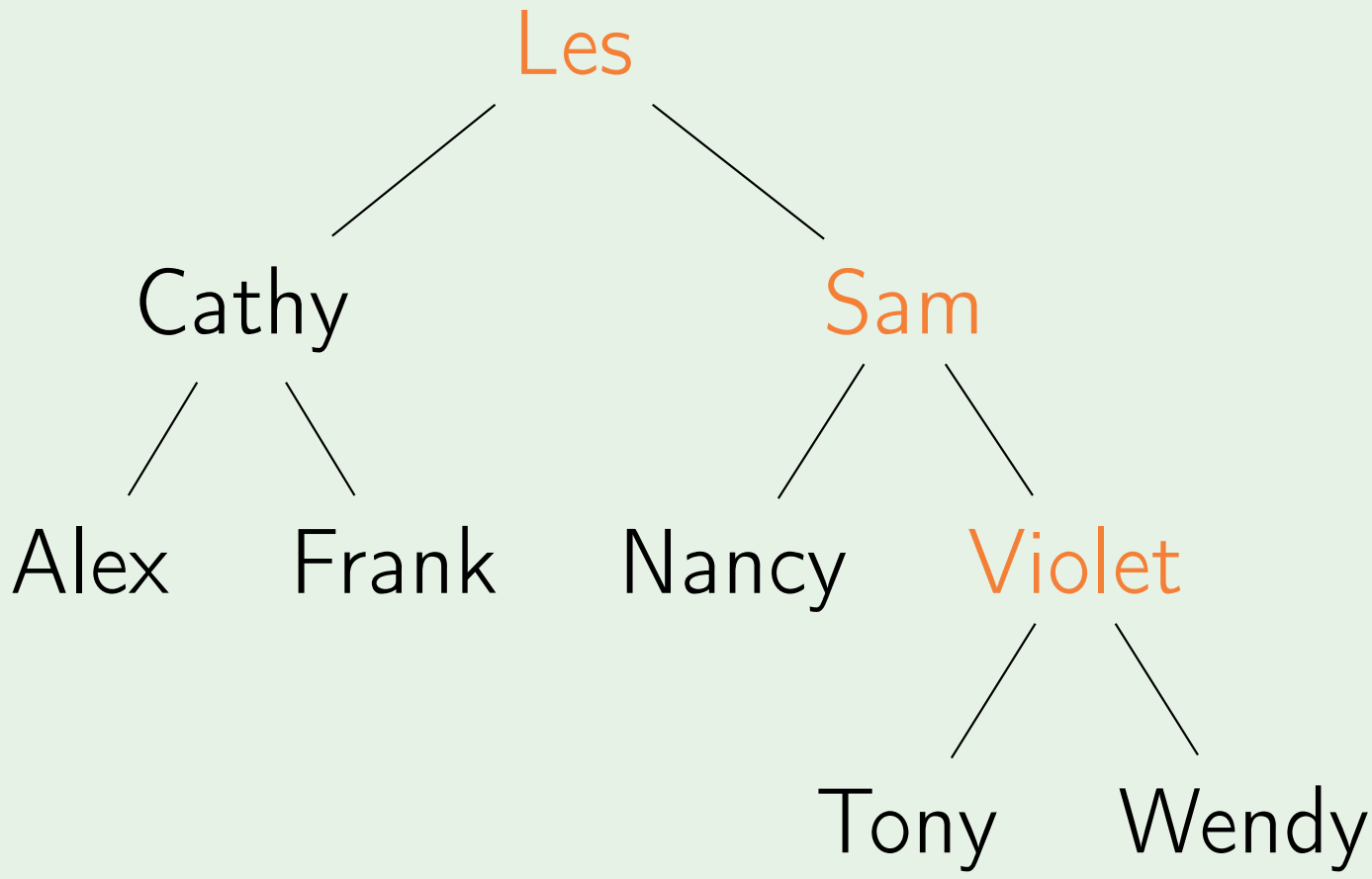
Output: Les Cathy Alex Frank Sam Nancy

PreOrderTraversal



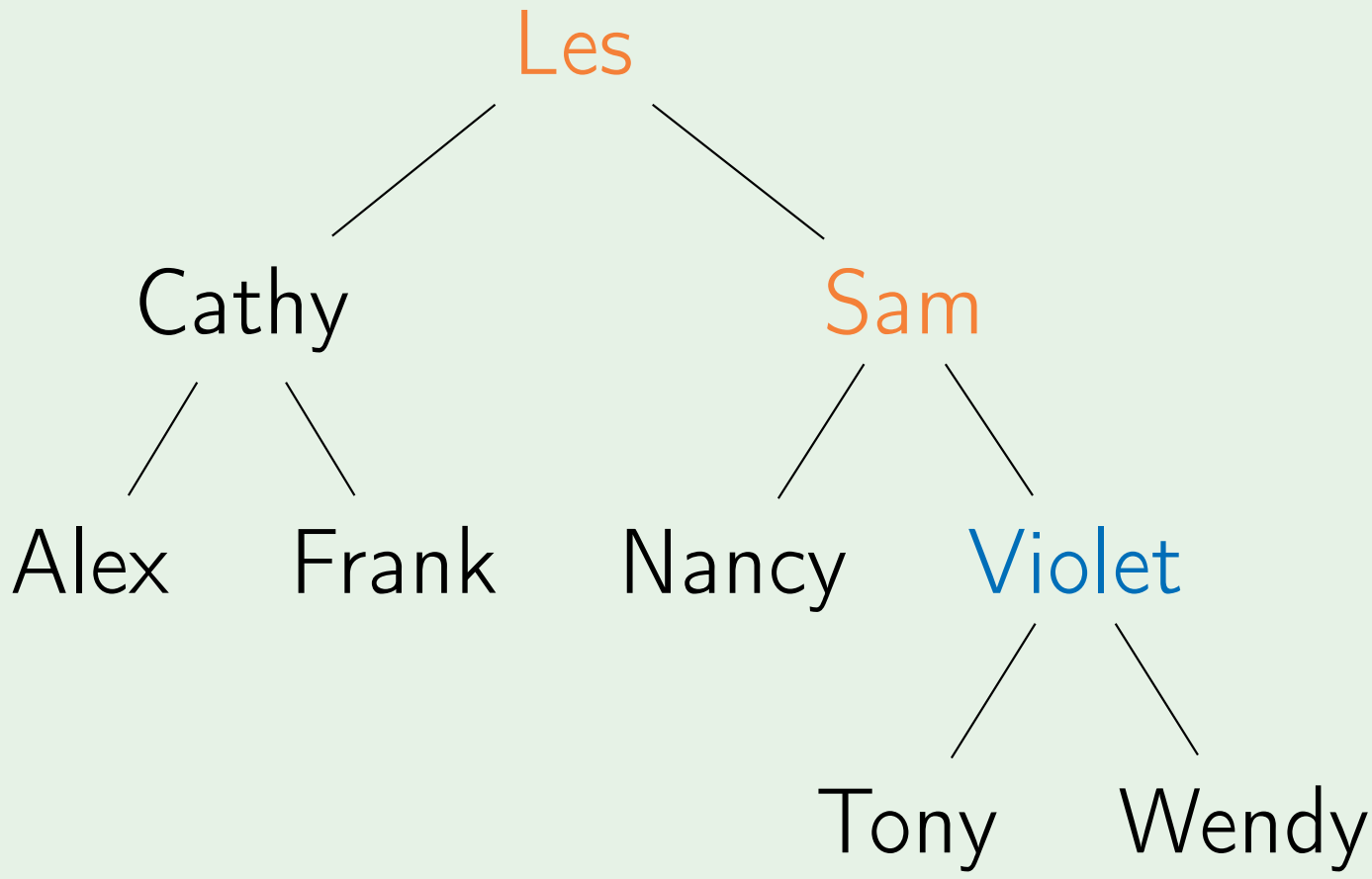
Output: Les Cathy Alex Frank Sam Nancy

PreOrderTraversal



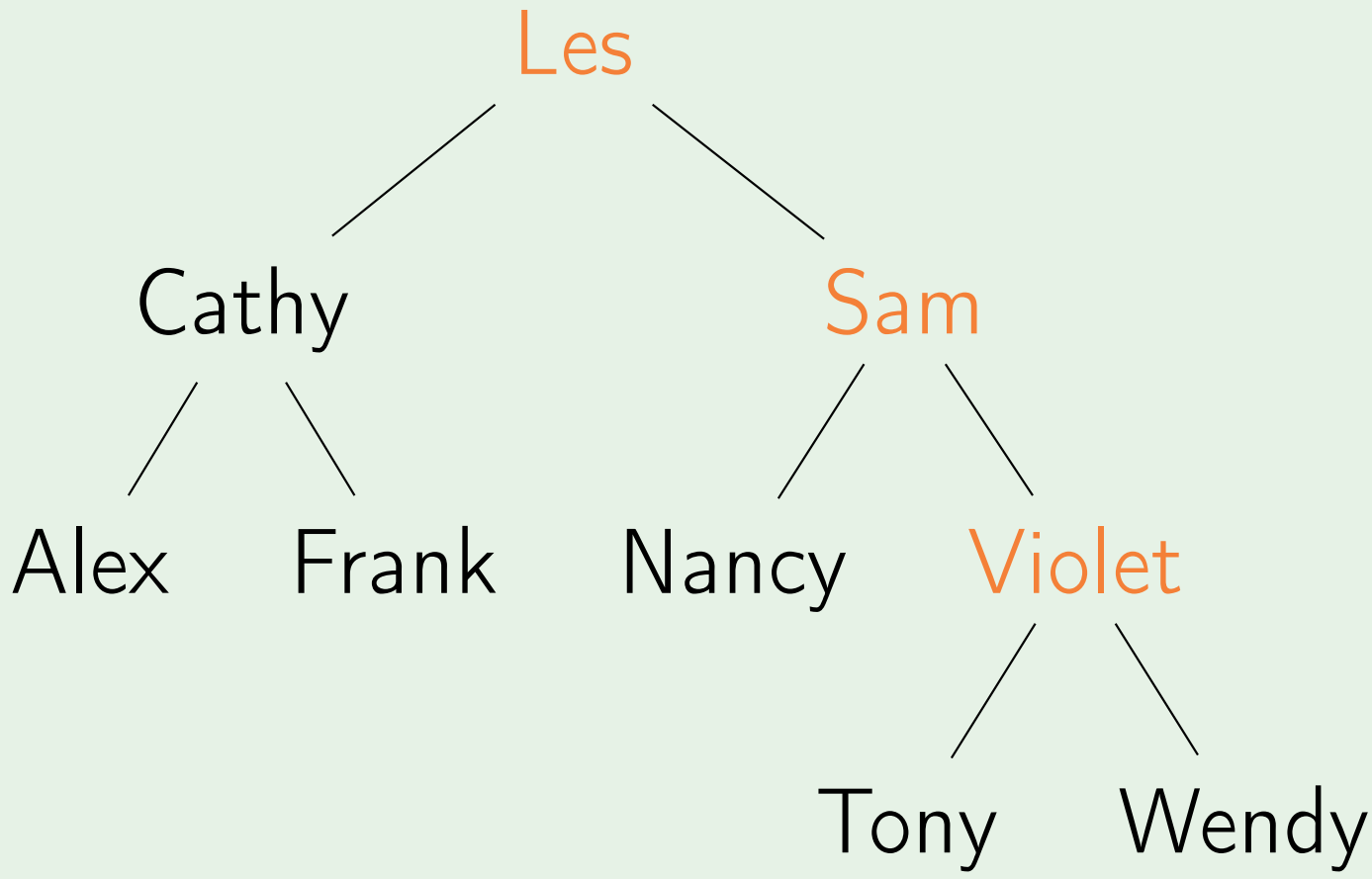
Output: Les Cathy Alex Frank Sam Nancy

PreOrderTraversal



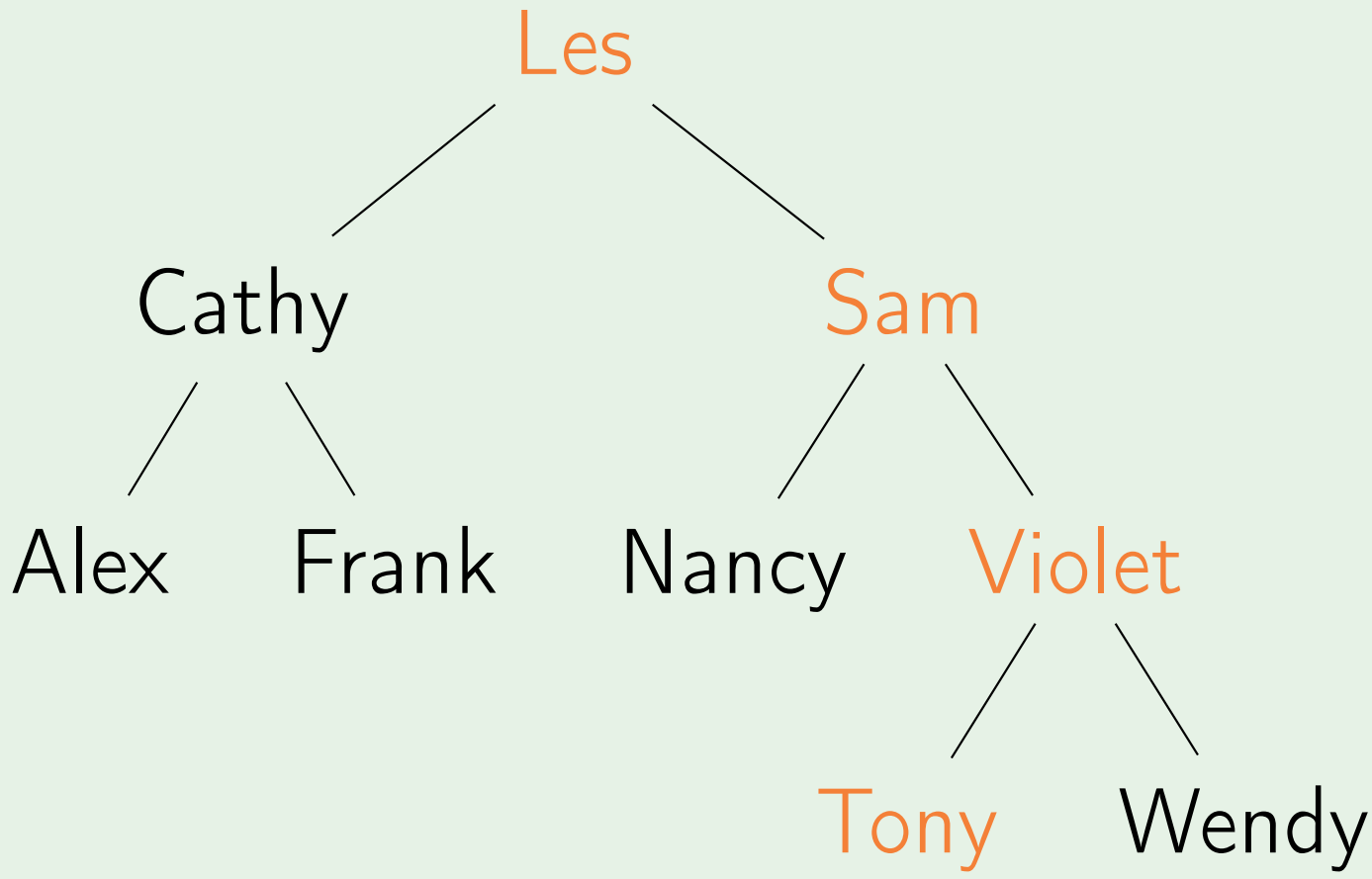
Output: Les Cathy Alex Frank Sam Nancy Violet

PreOrderTraversal



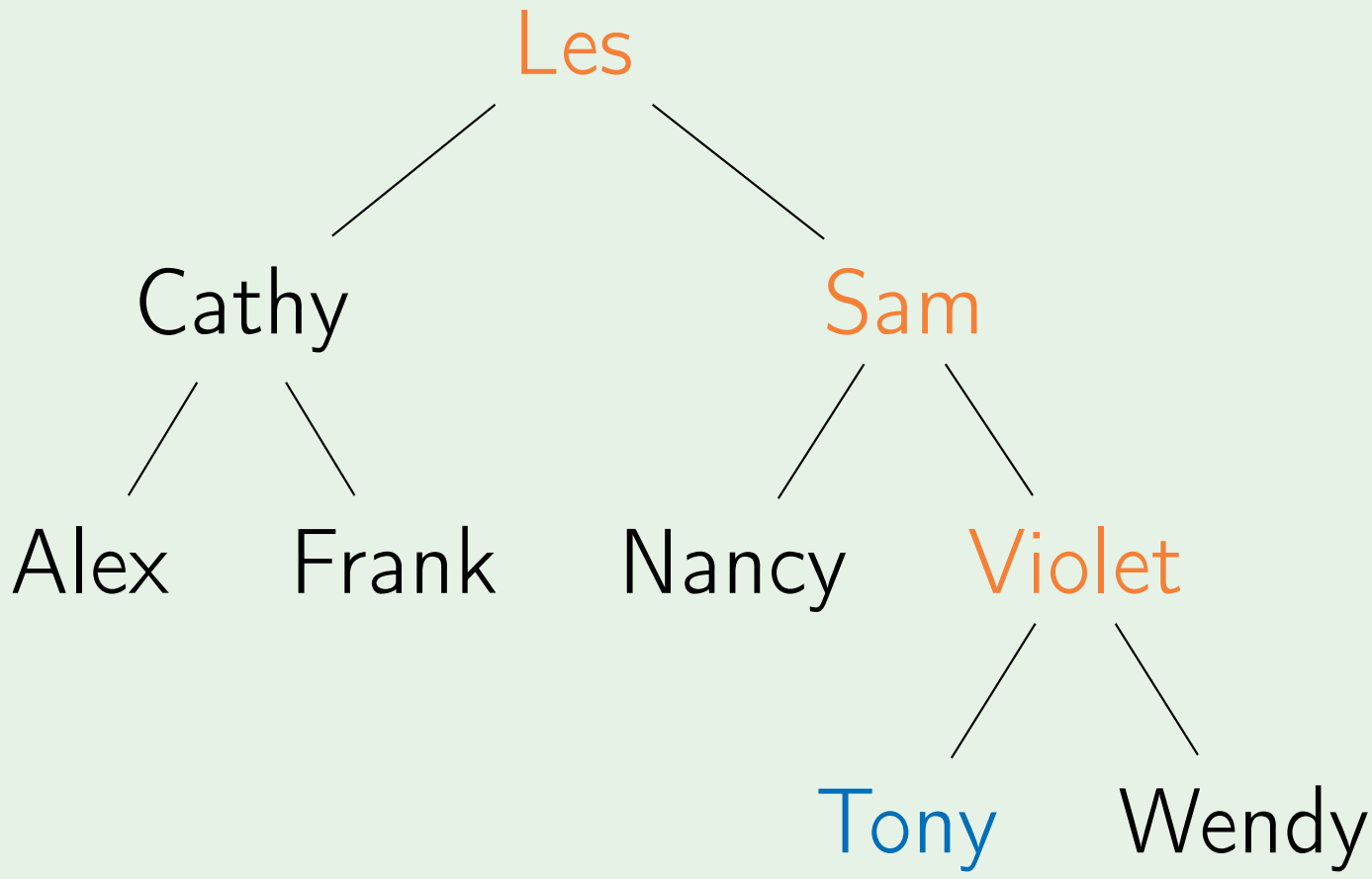
Output: Les Cathy Alex Frank Sam Nancy Violet

PreOrderTraversal



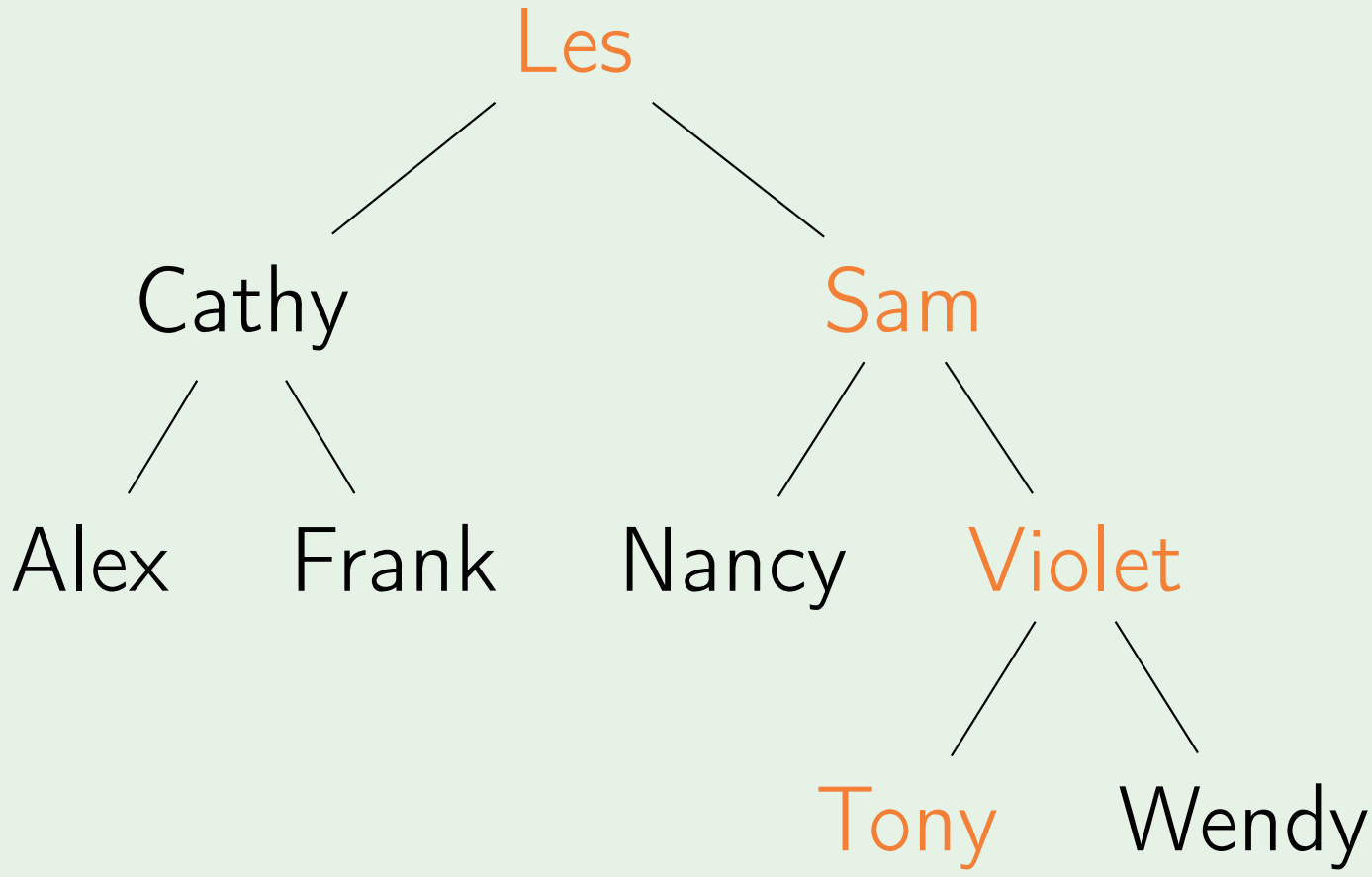
Output: Les Cathy Alex Frank Sam Nancy Violet

PreOrderTraversal



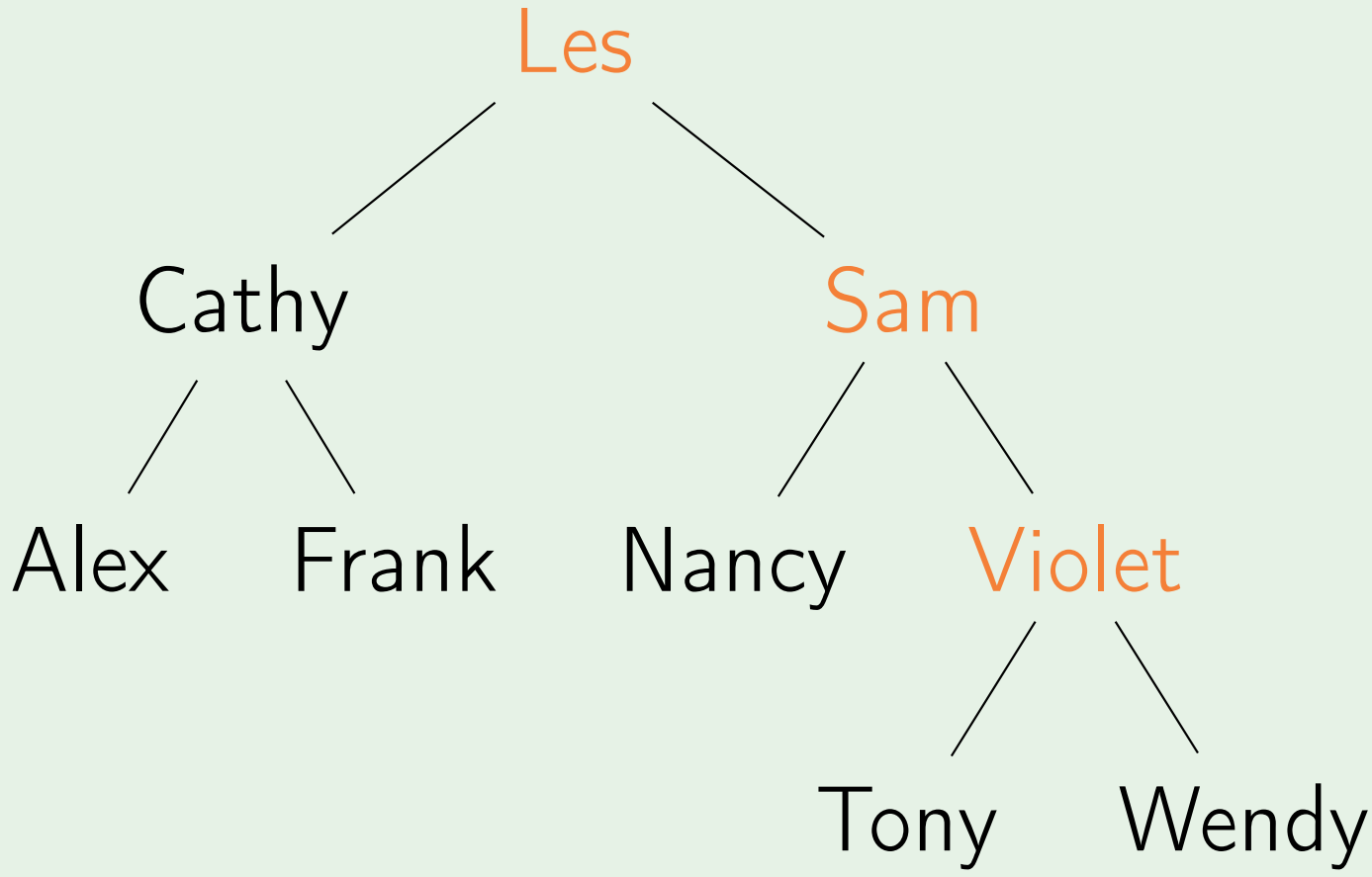
Output: Les Cathy Alex Frank Sam Nancy Violet Tony

PreOrderTraversal



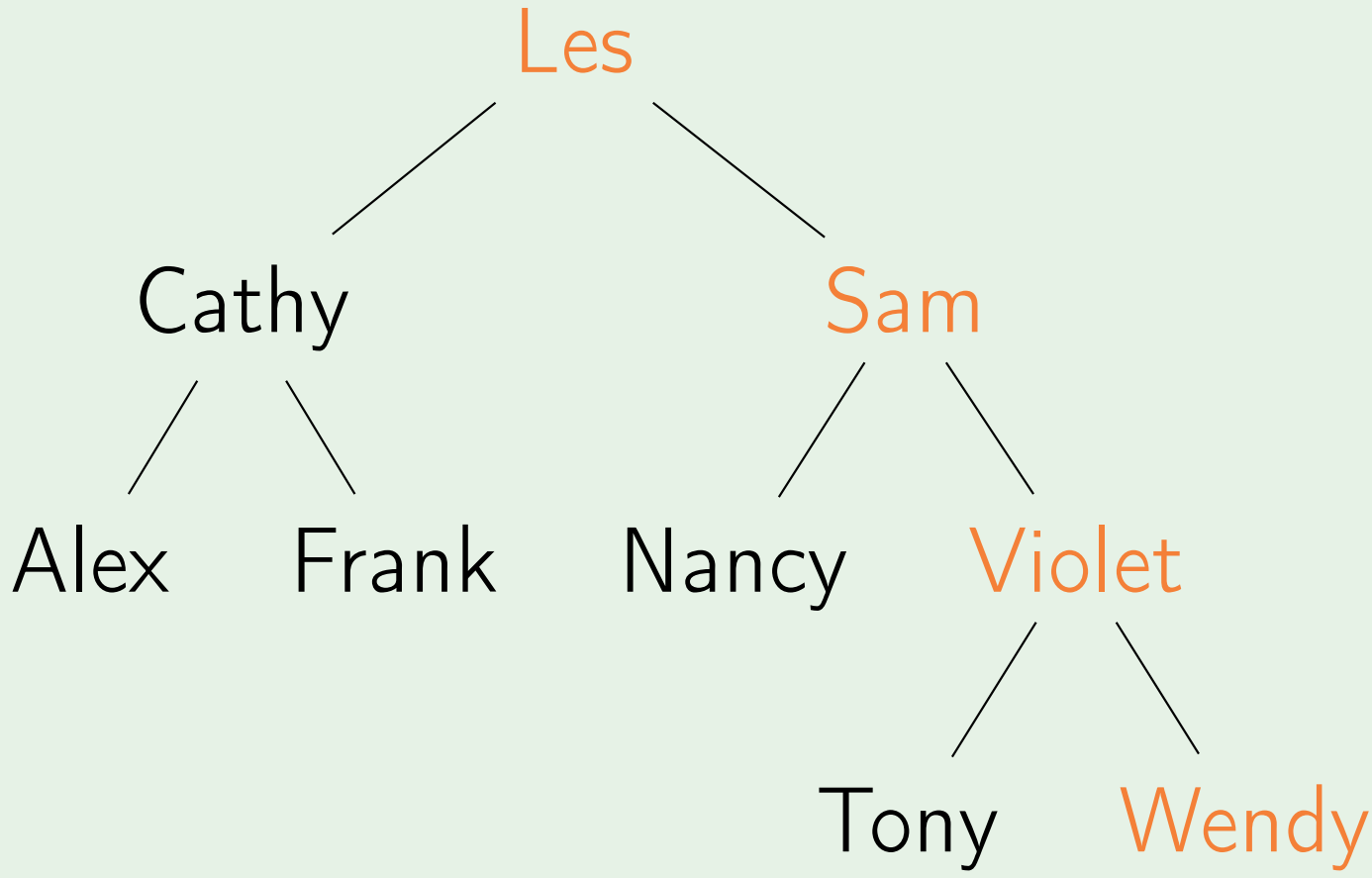
Output: Les Cathy Alex Frank Sam Nancy Violet Tony

PreOrderTraversal



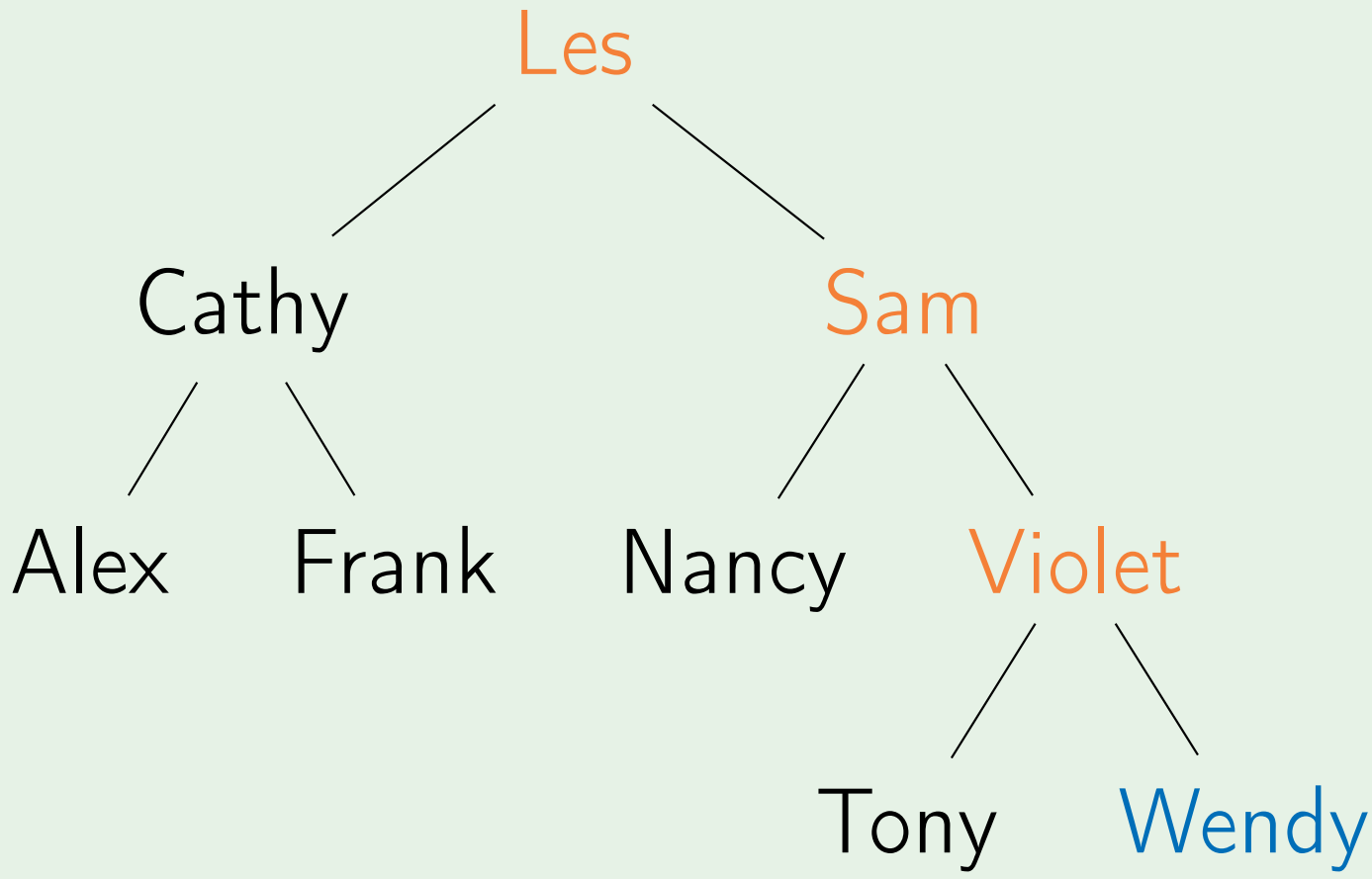
Output: Les Cathy Alex Frank Sam Nancy Violet Tony

PreOrderTraversal



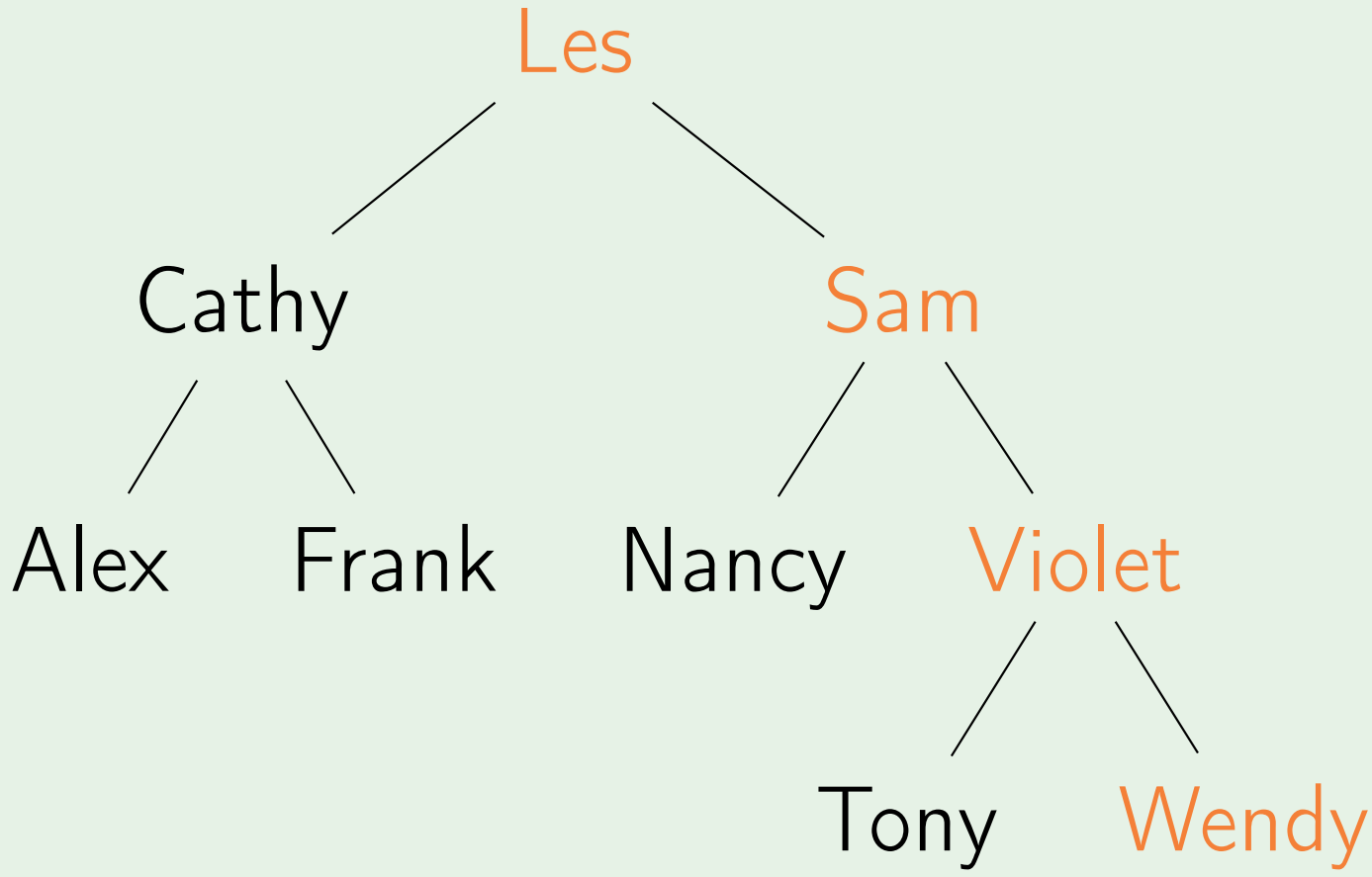
Output: Les Cathy Alex Frank Sam Nancy Violet Tony

PreOrderTraversal



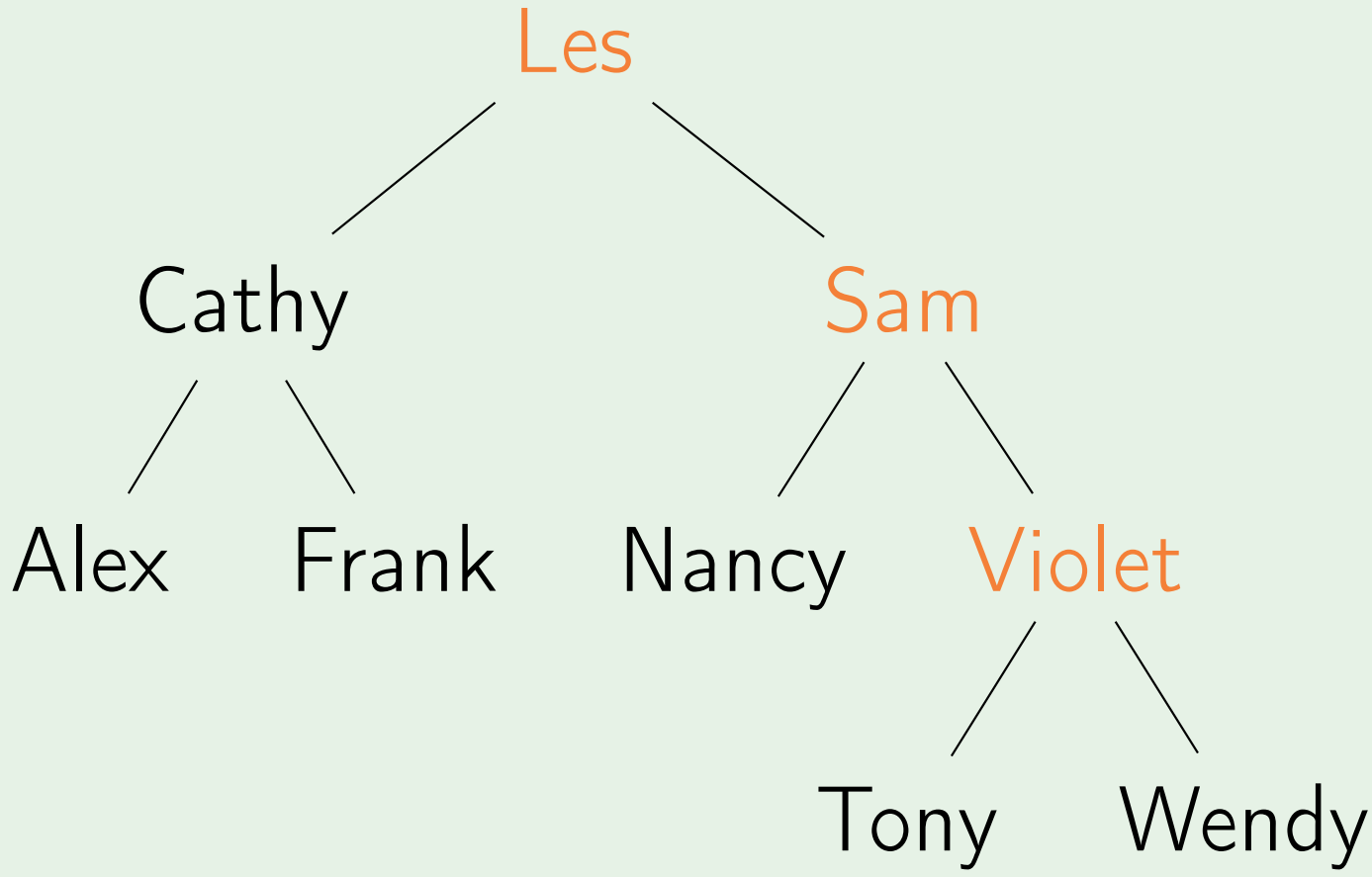
Output: Les Cathy Alex Frank Sam Nancy Violet Tony Wendy

PreOrderTraversal



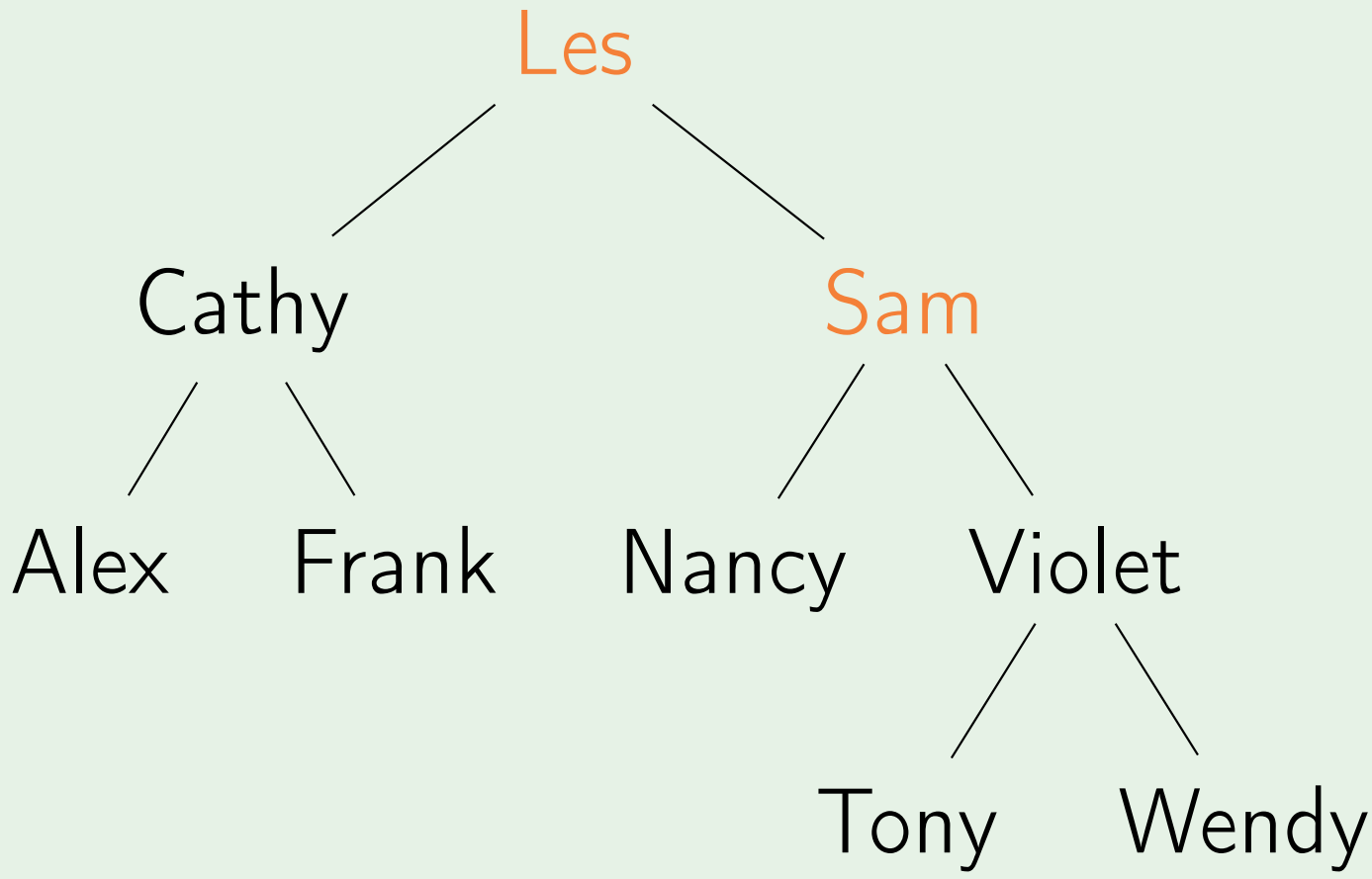
Output: Les Cathy Alex Frank Sam Nancy Violet Tony Wendy

PreOrderTraversal



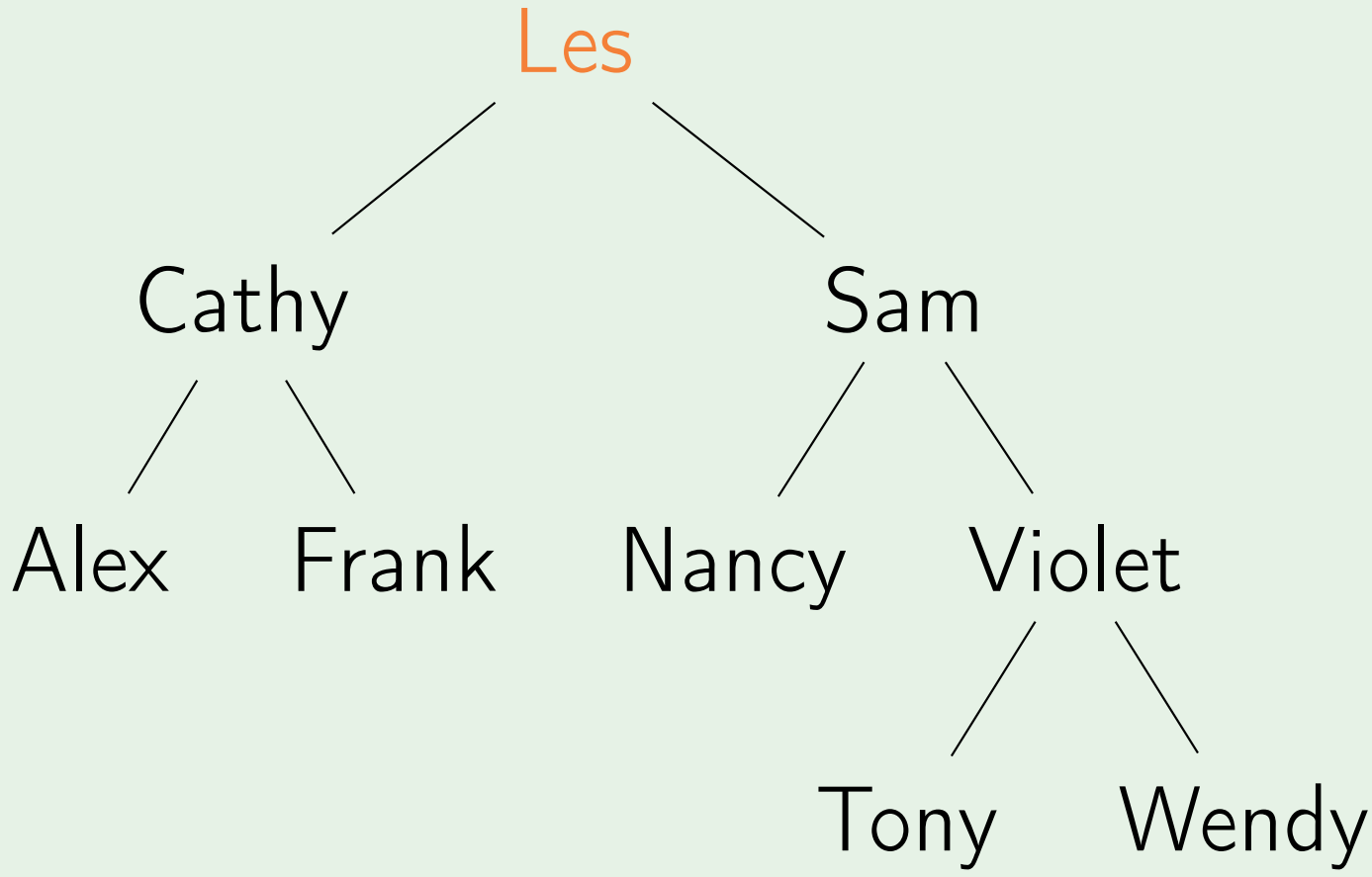
Output: Les Cathy Alex Frank Sam Nancy Violet Tony Wendy

PreOrderTraversal



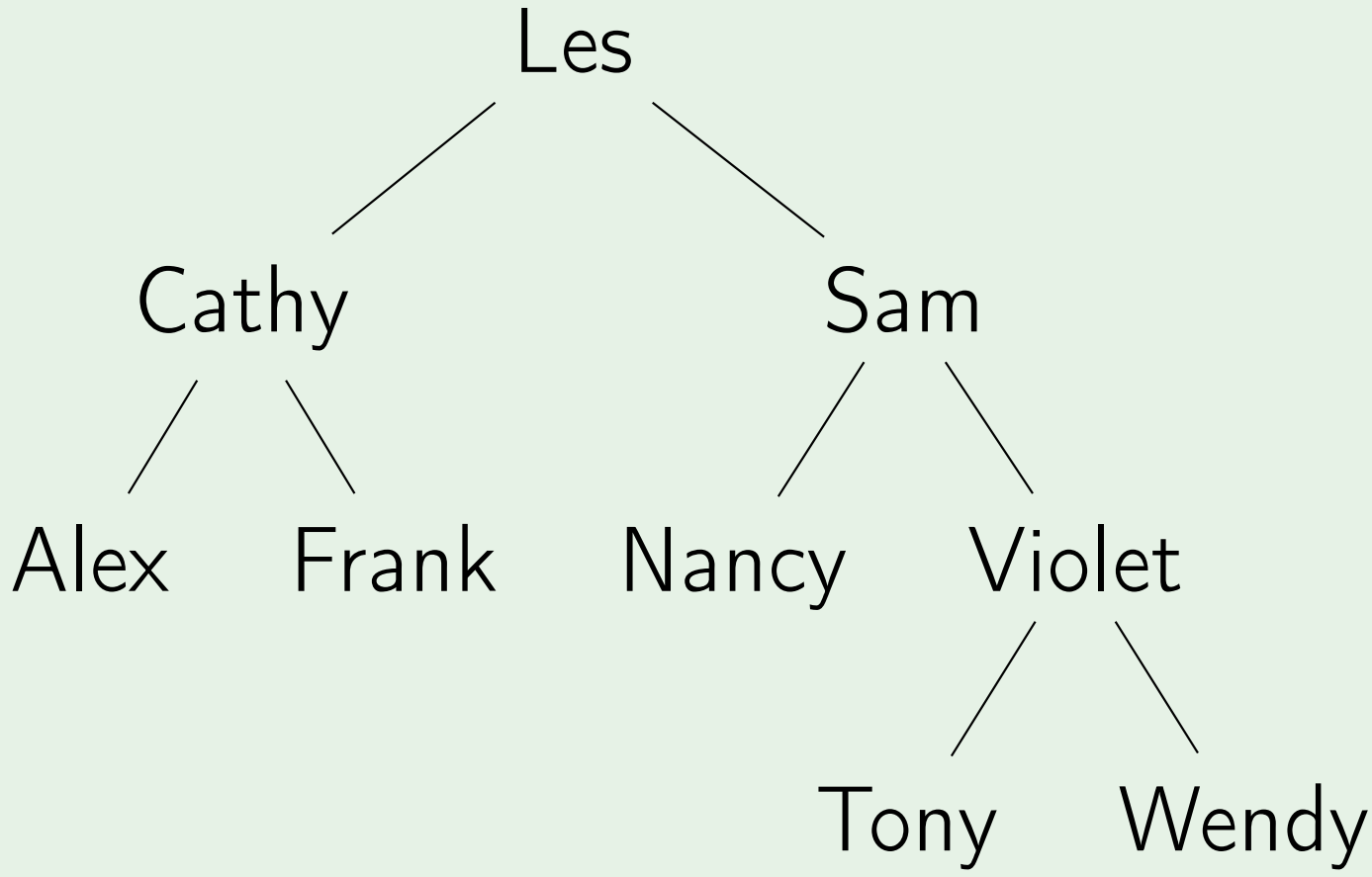
Output: Les Cathy Alex Frank Sam Nancy Violet Tony Wendy

PreOrderTraversal



Output: Les Cathy Alex Frank Sam Nancy Violet Tony Wendy

PreOrderTraversal

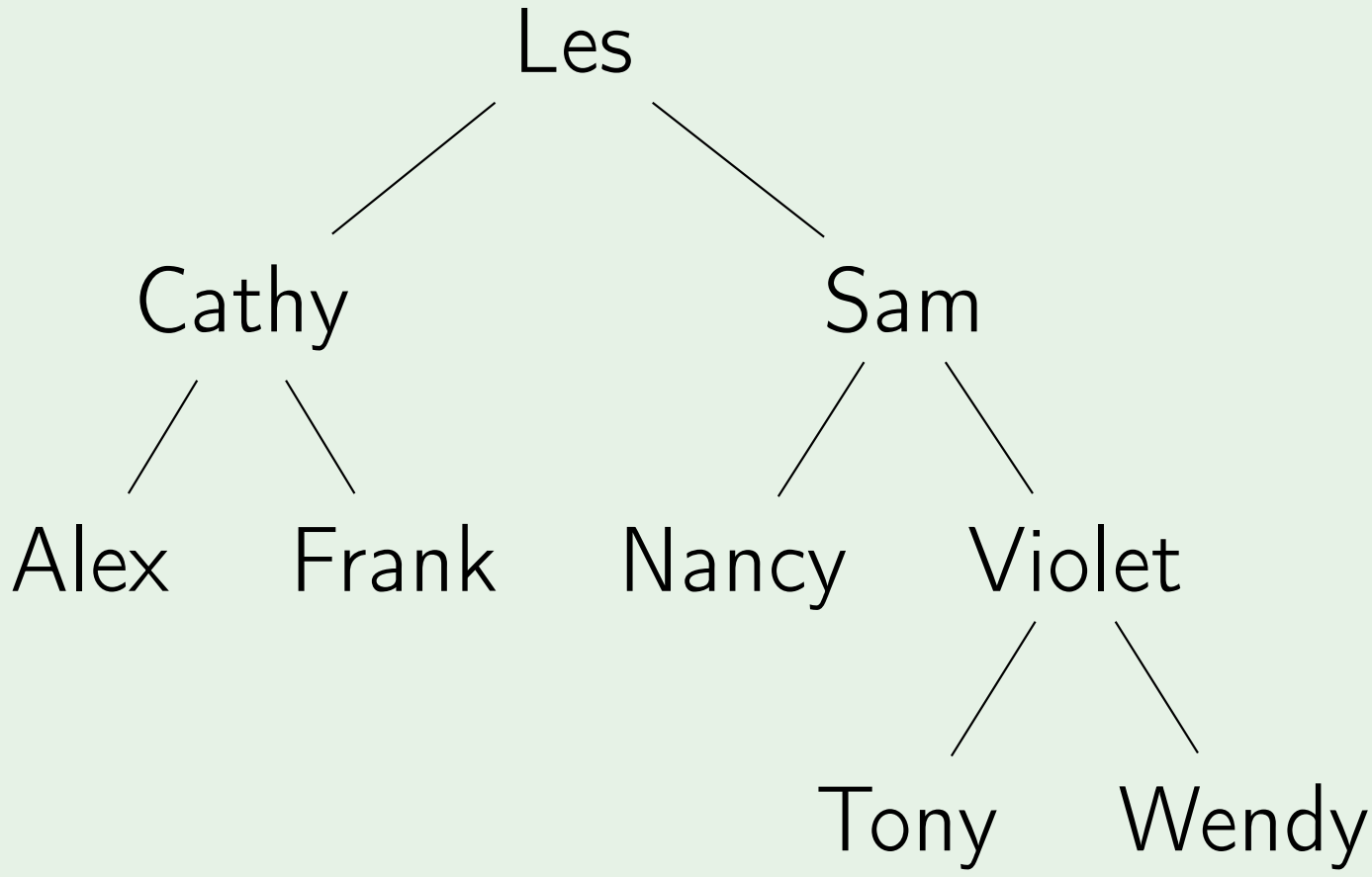


Output: Les Cathy Alex Frank Sam Nancy Violet Tony Wendy

Depth-first

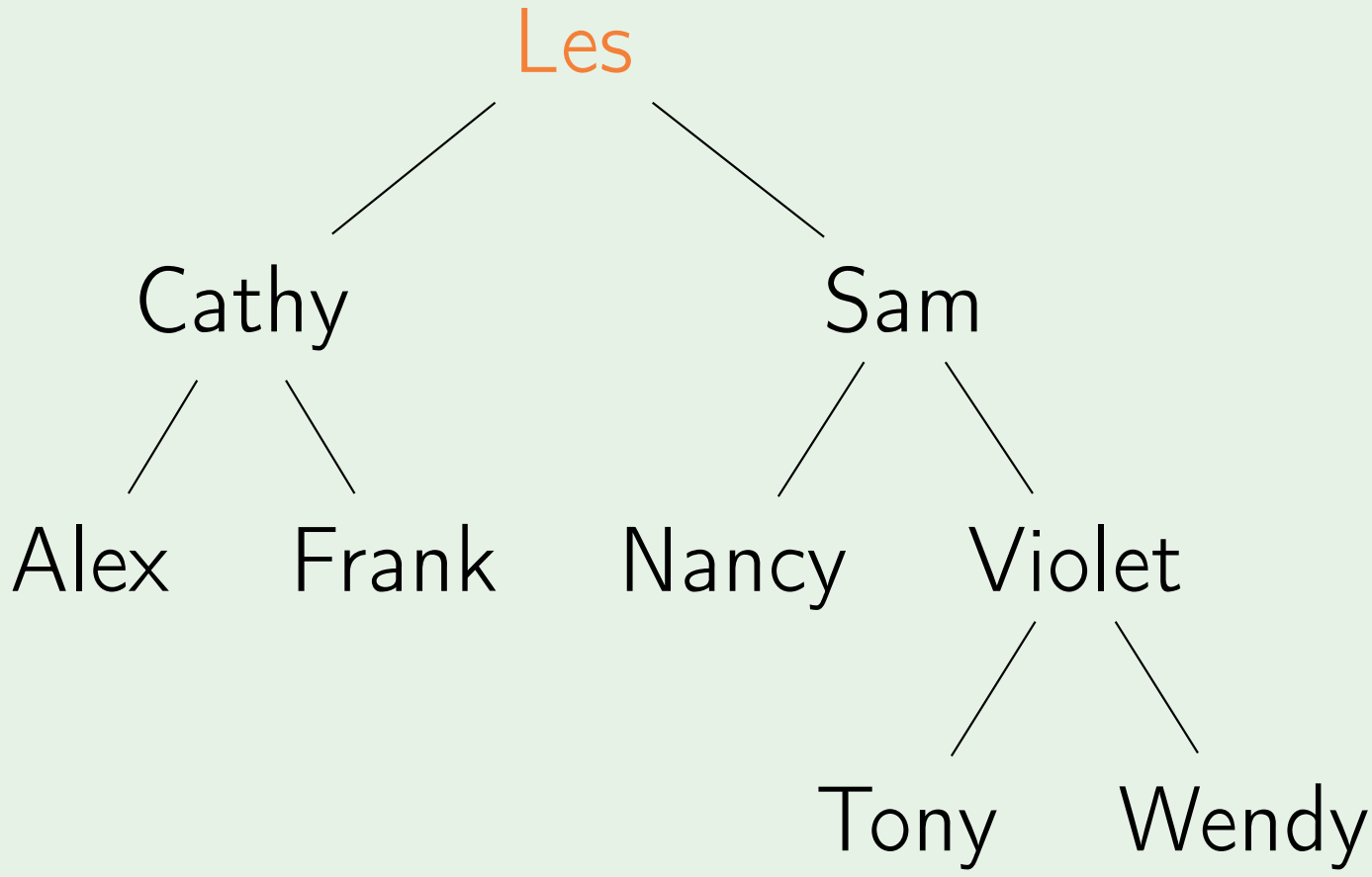
```
PostOrderTraversal(tree)
if tree = nil:
   return
PostOrderTraversal(tree.left)
PostOrderTraversal(tree.right)
Print(tree.key)
```

PostOrderTraversal



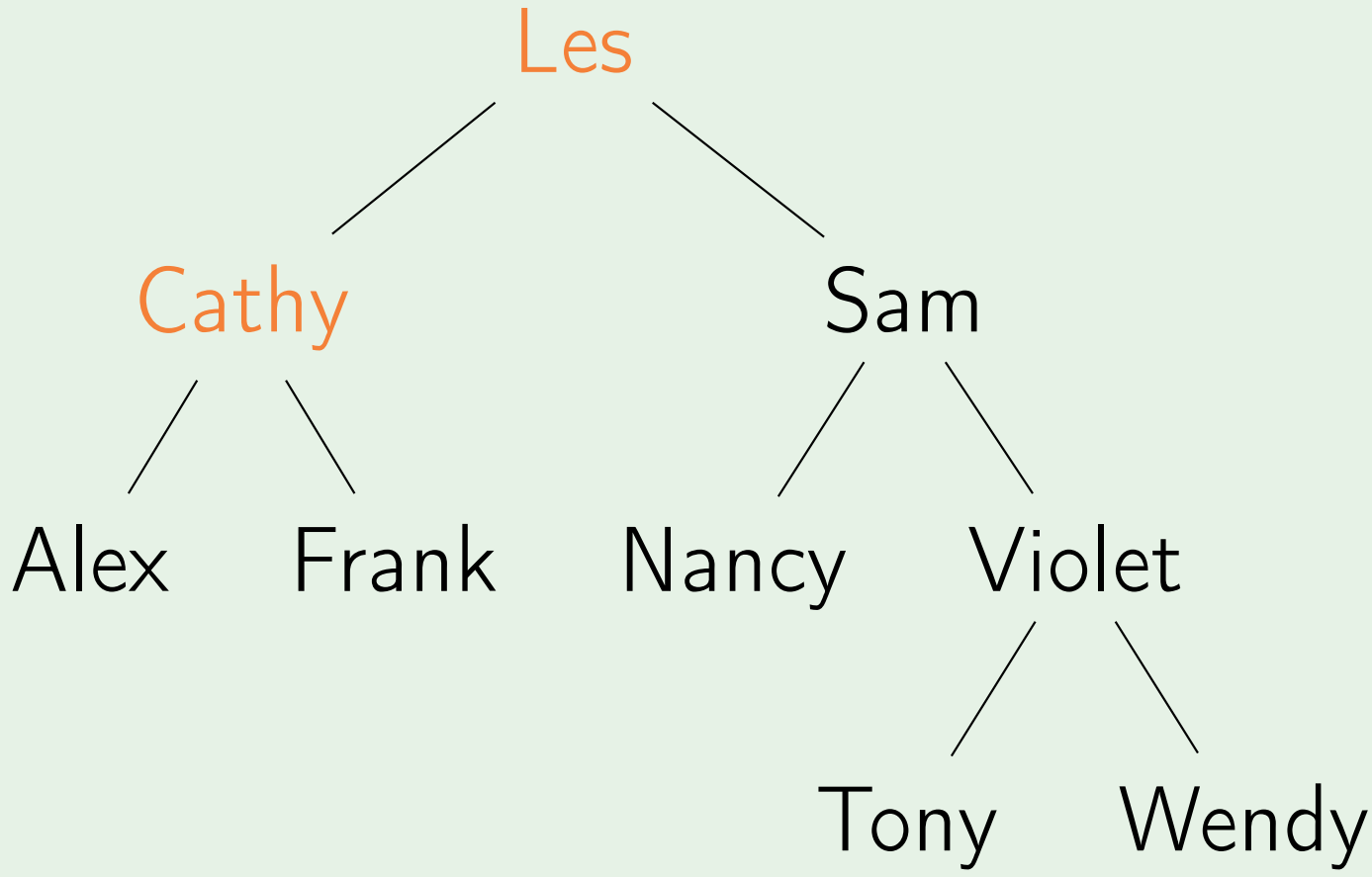
Output:

PostOrderTraversal



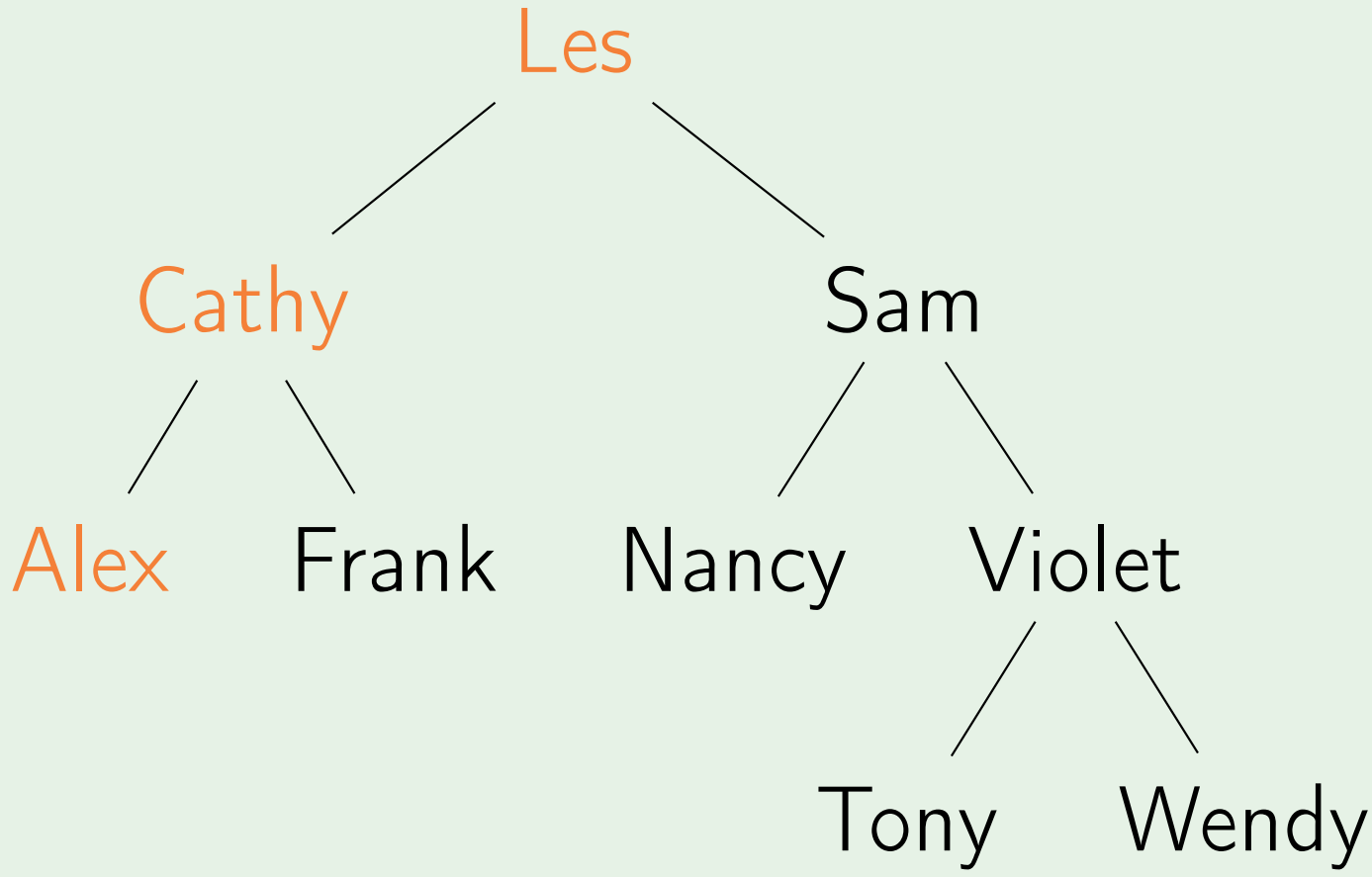
Output:

PostOrderTraversal



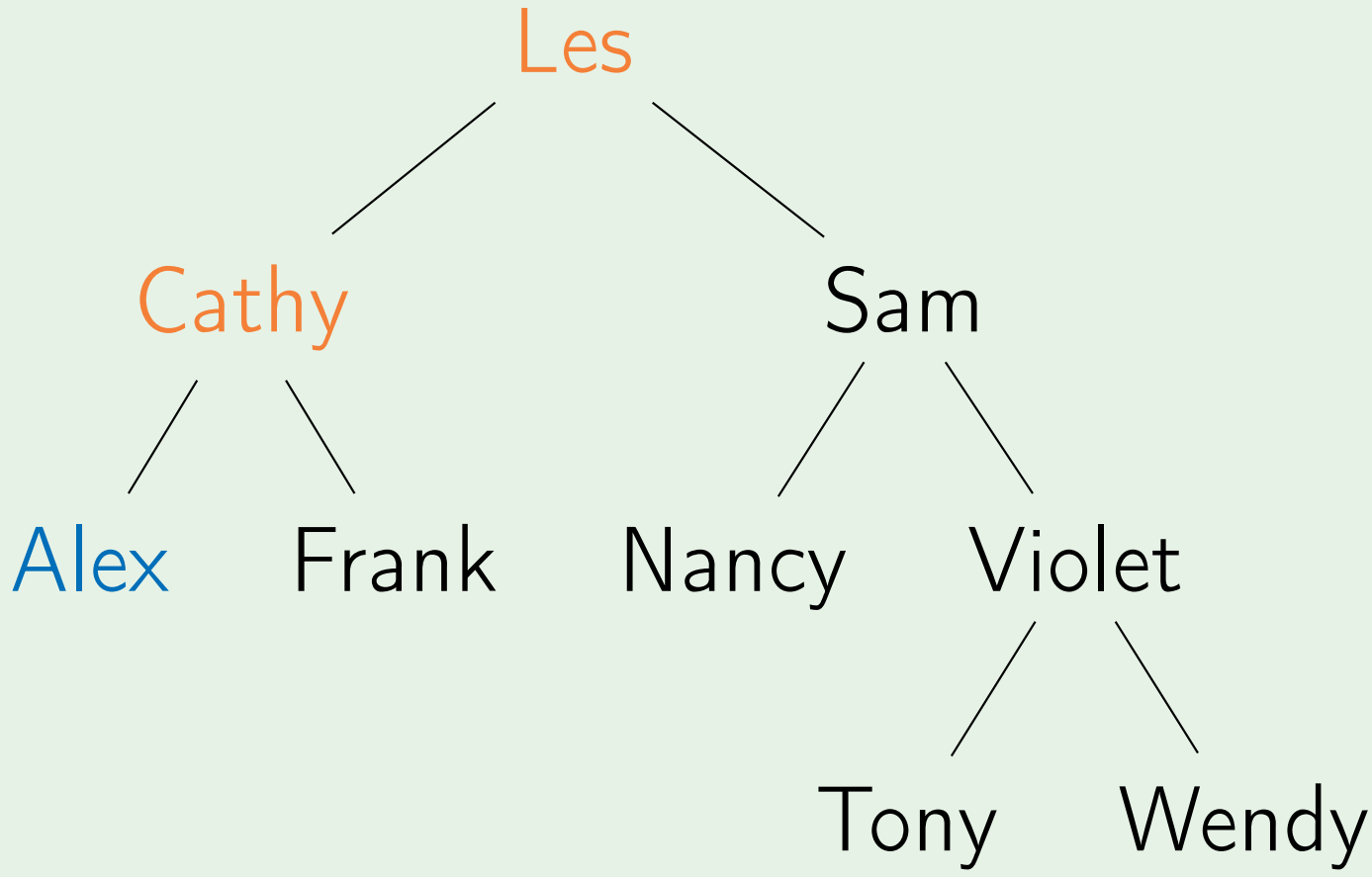
Output:

PostOrderTraversal



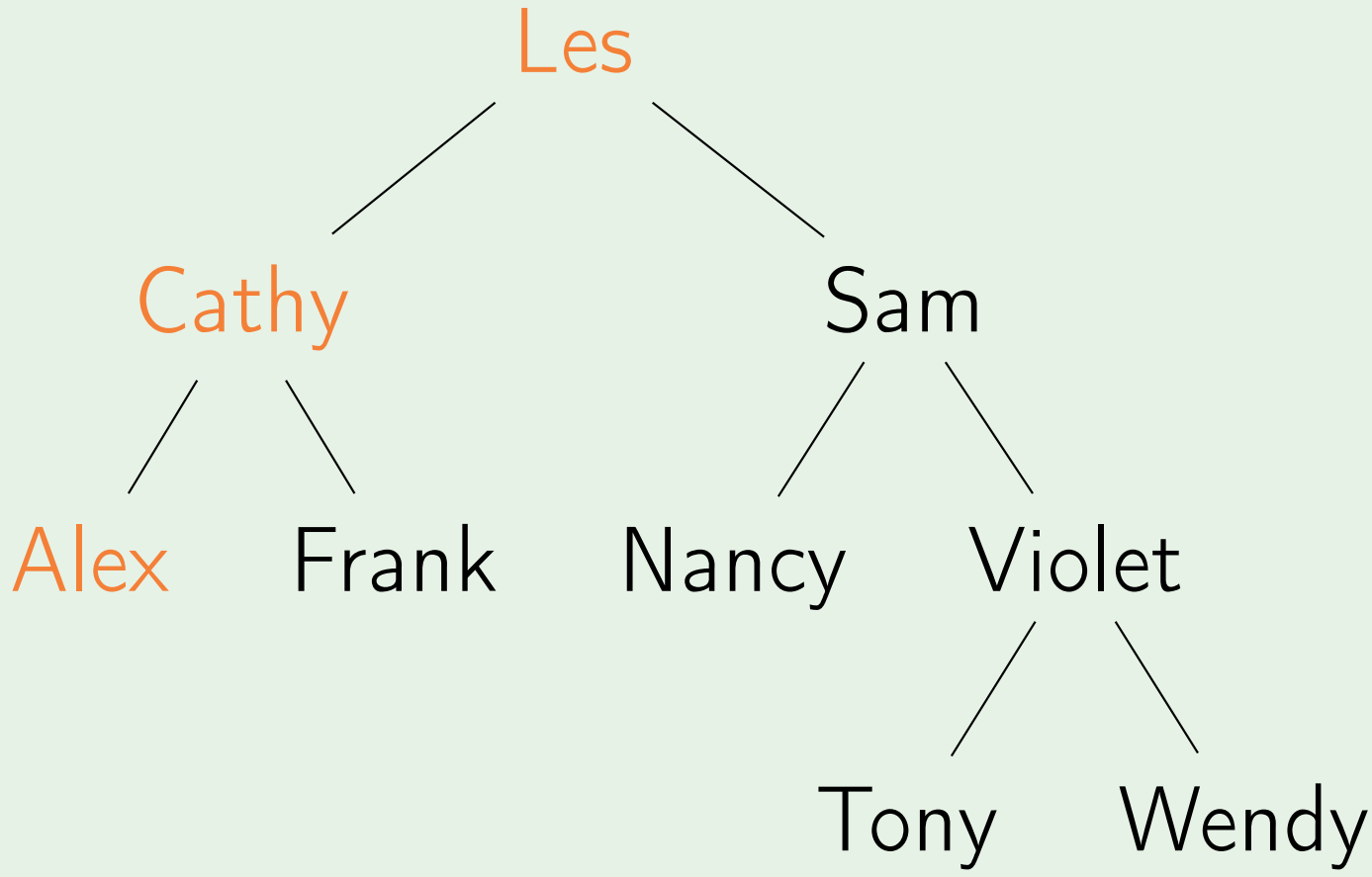
Output:

PostOrderTraversal



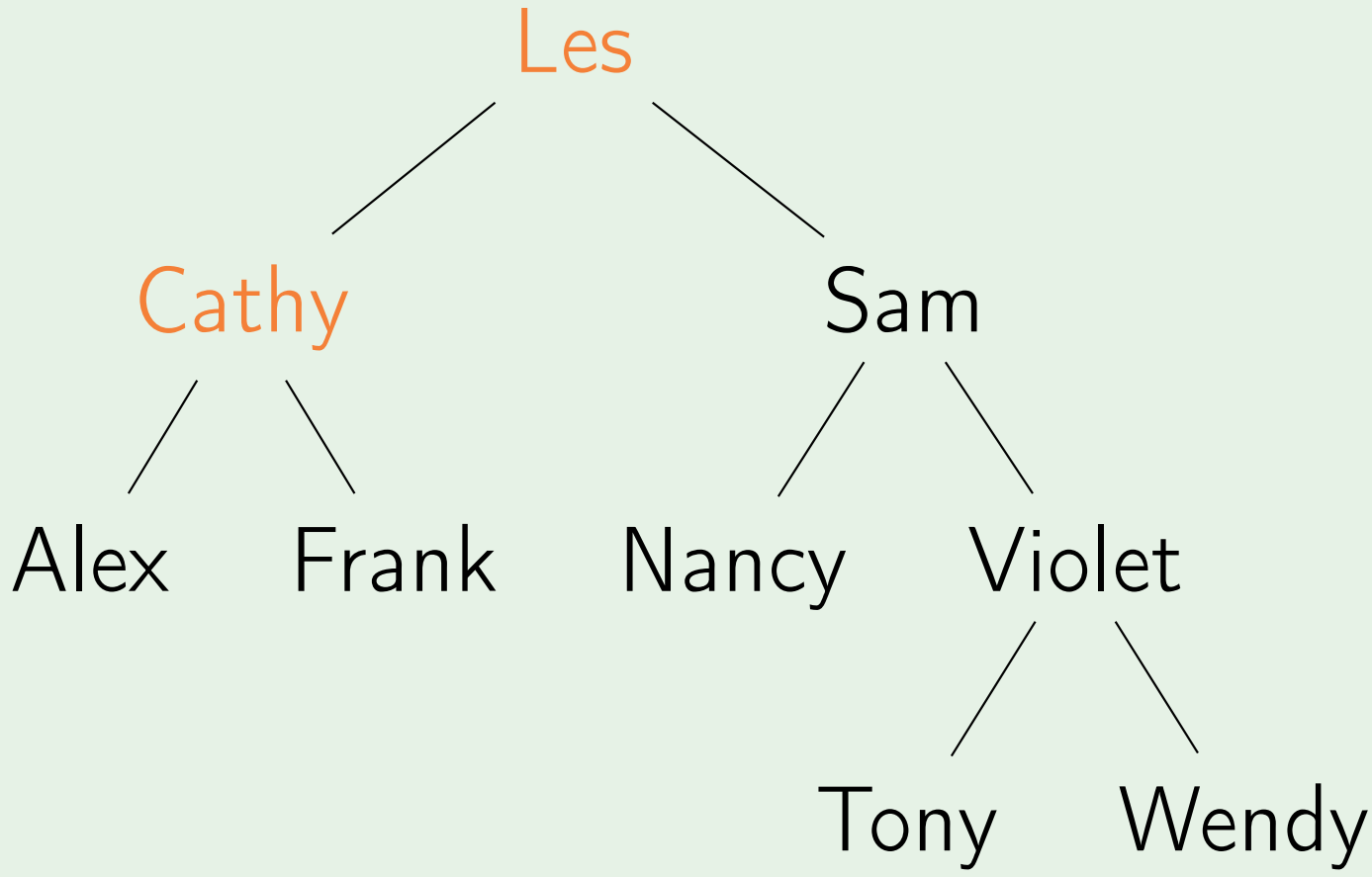
Output: Alex

PostOrderTraversal



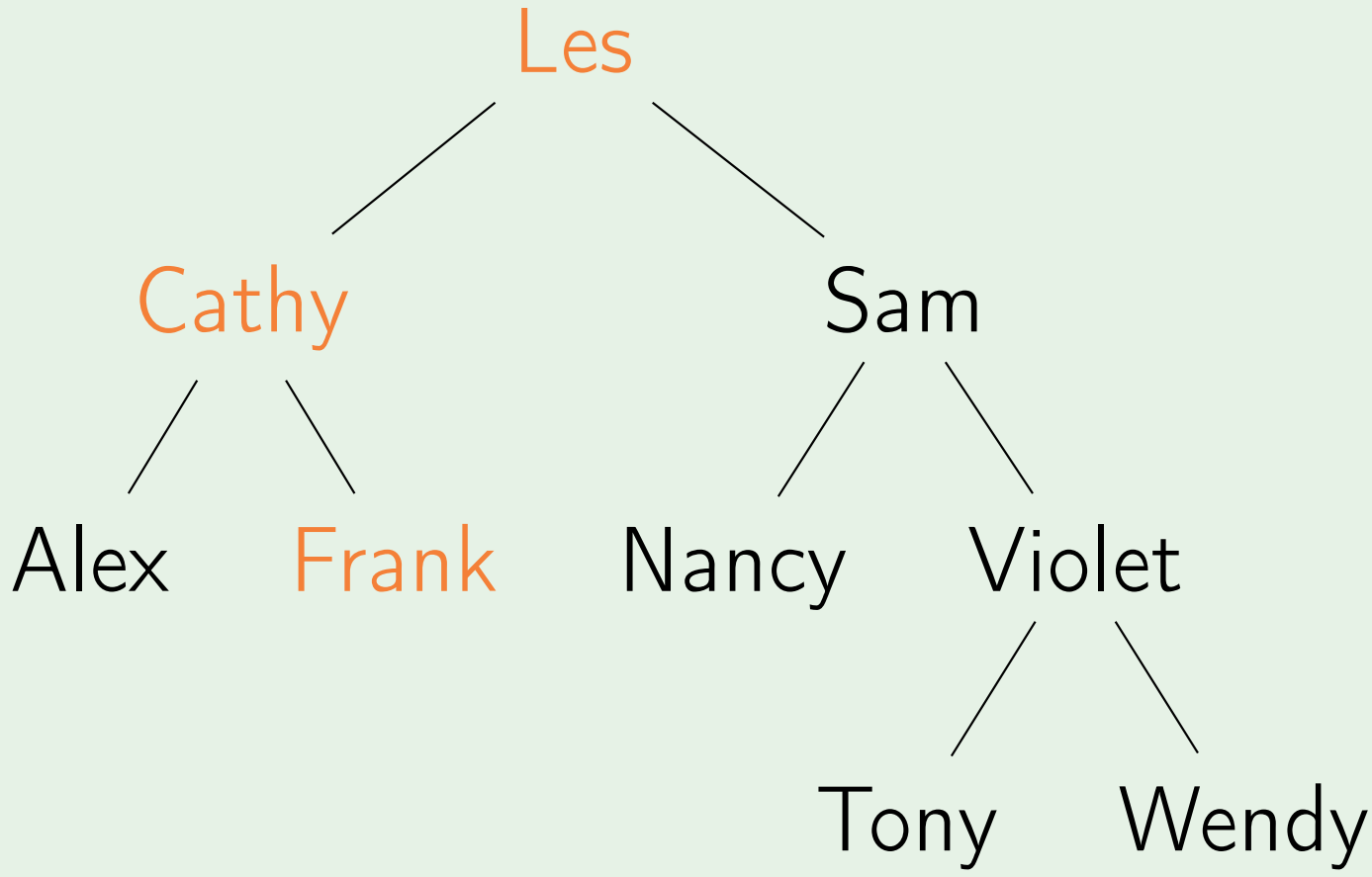
Output: Alex

PostOrderTraversal



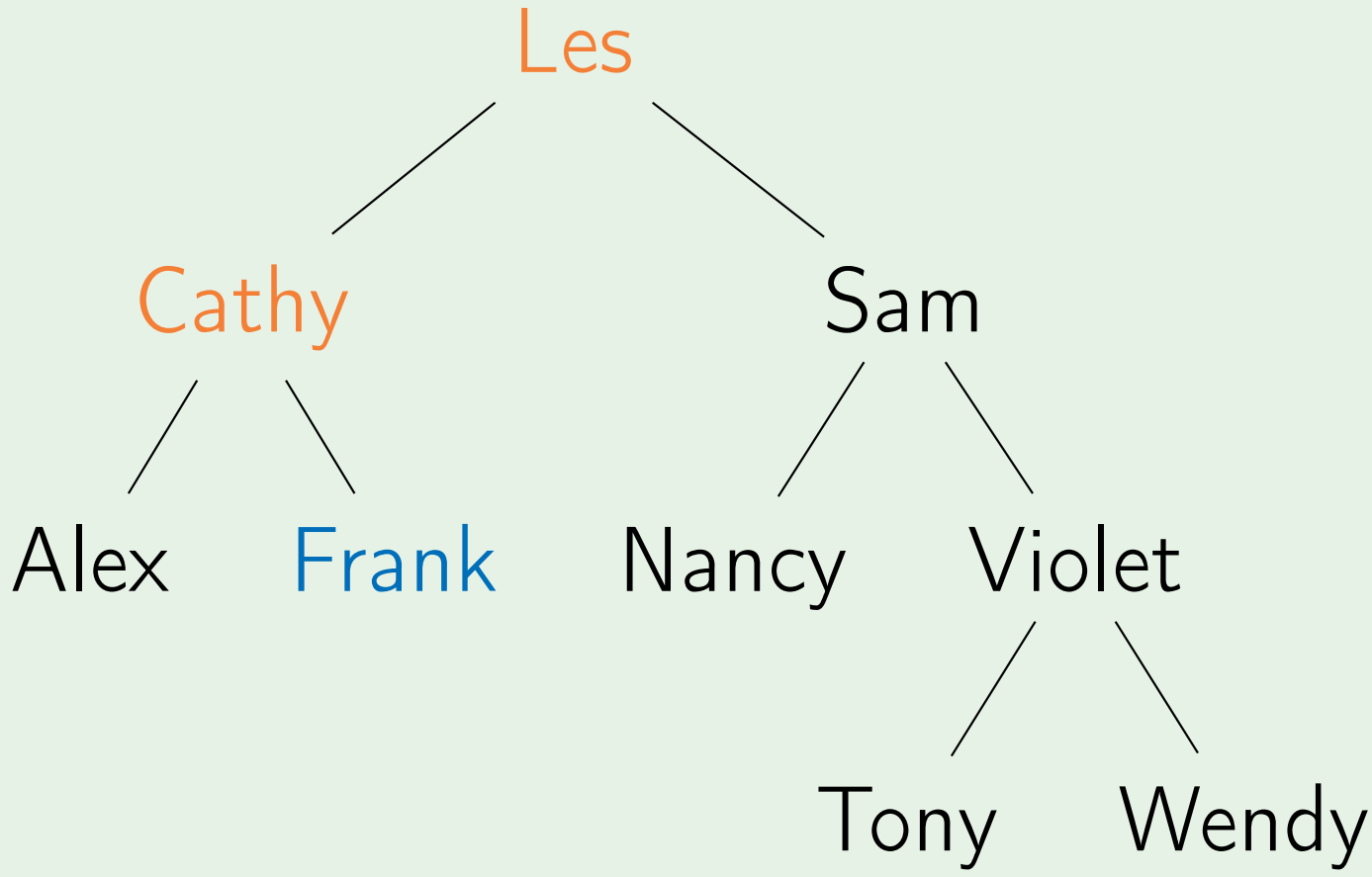
Output: Alex

PostOrderTraversal



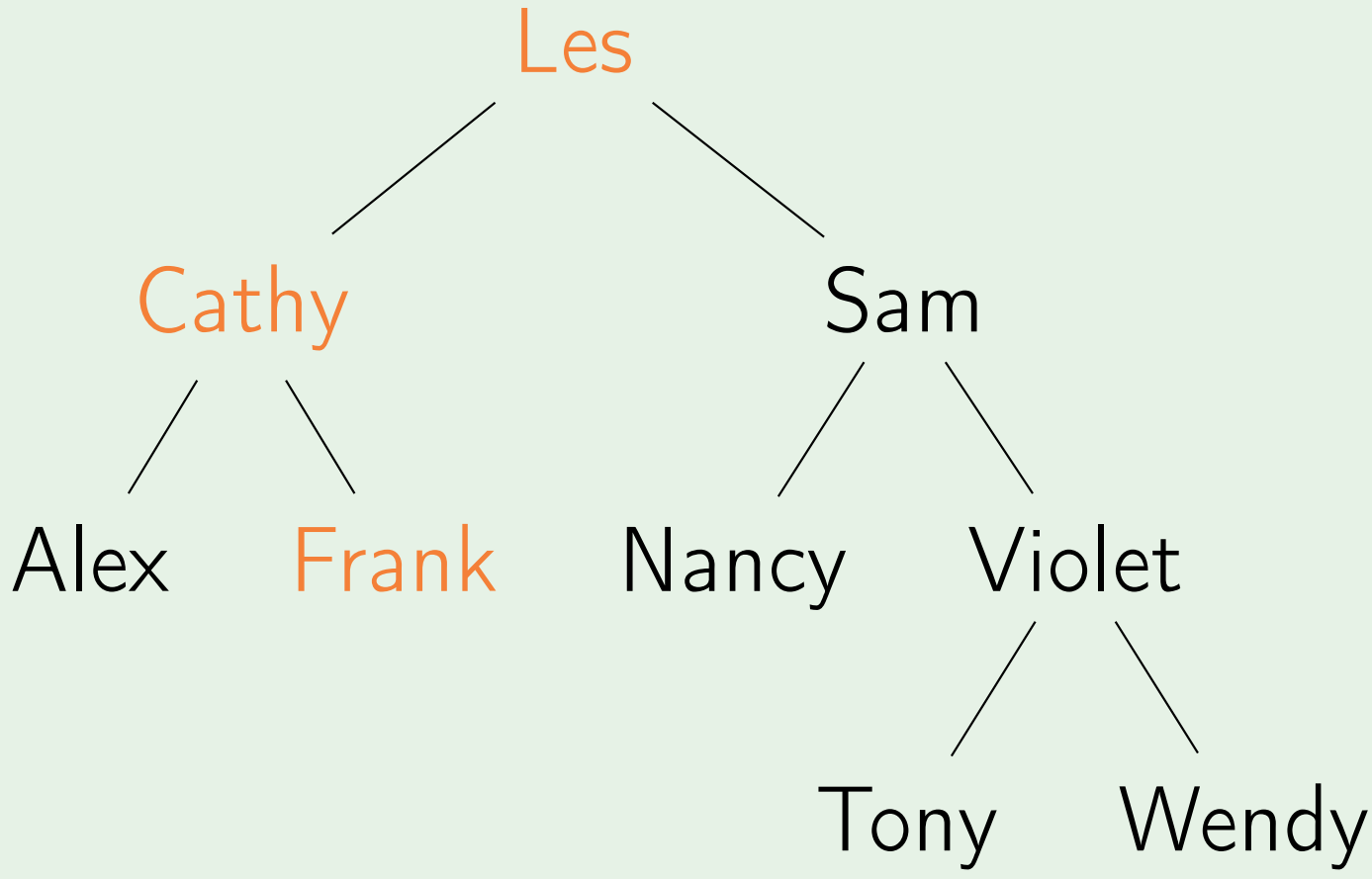
Output: Alex

PostOrderTraversal



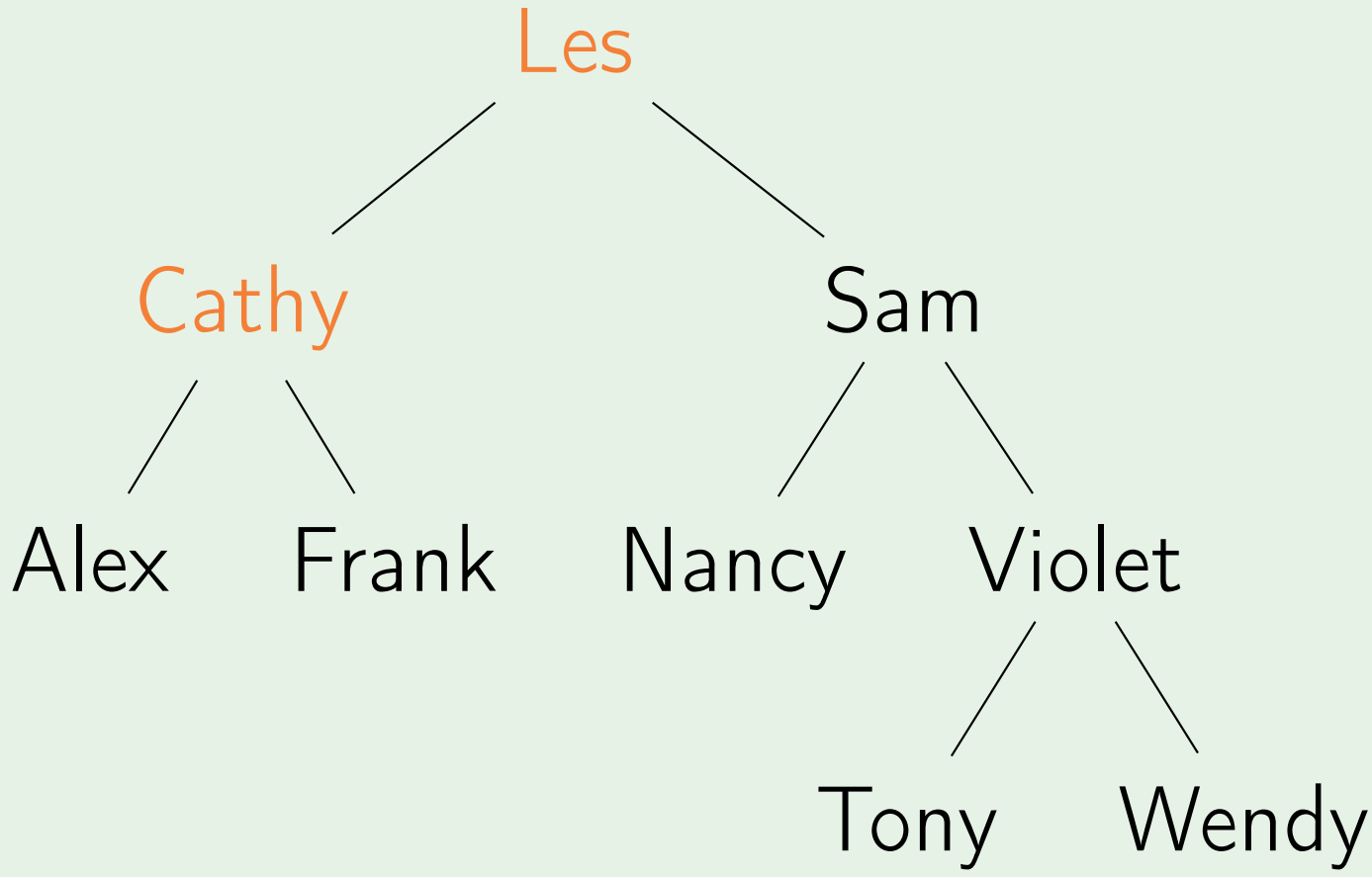
Output: Alex Frank

PostOrderTraversal



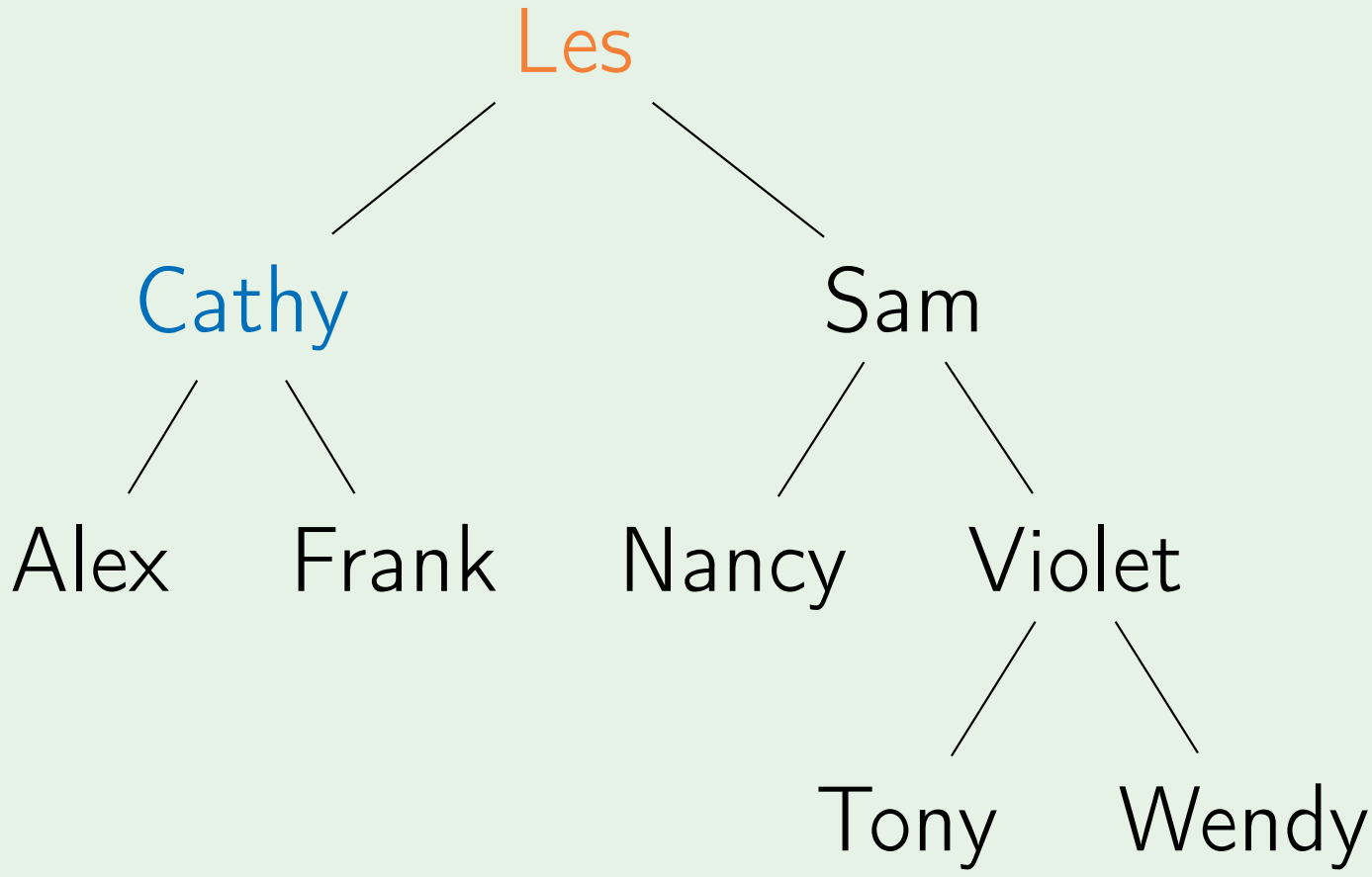
Output: Alex Frank

PostOrderTraversal



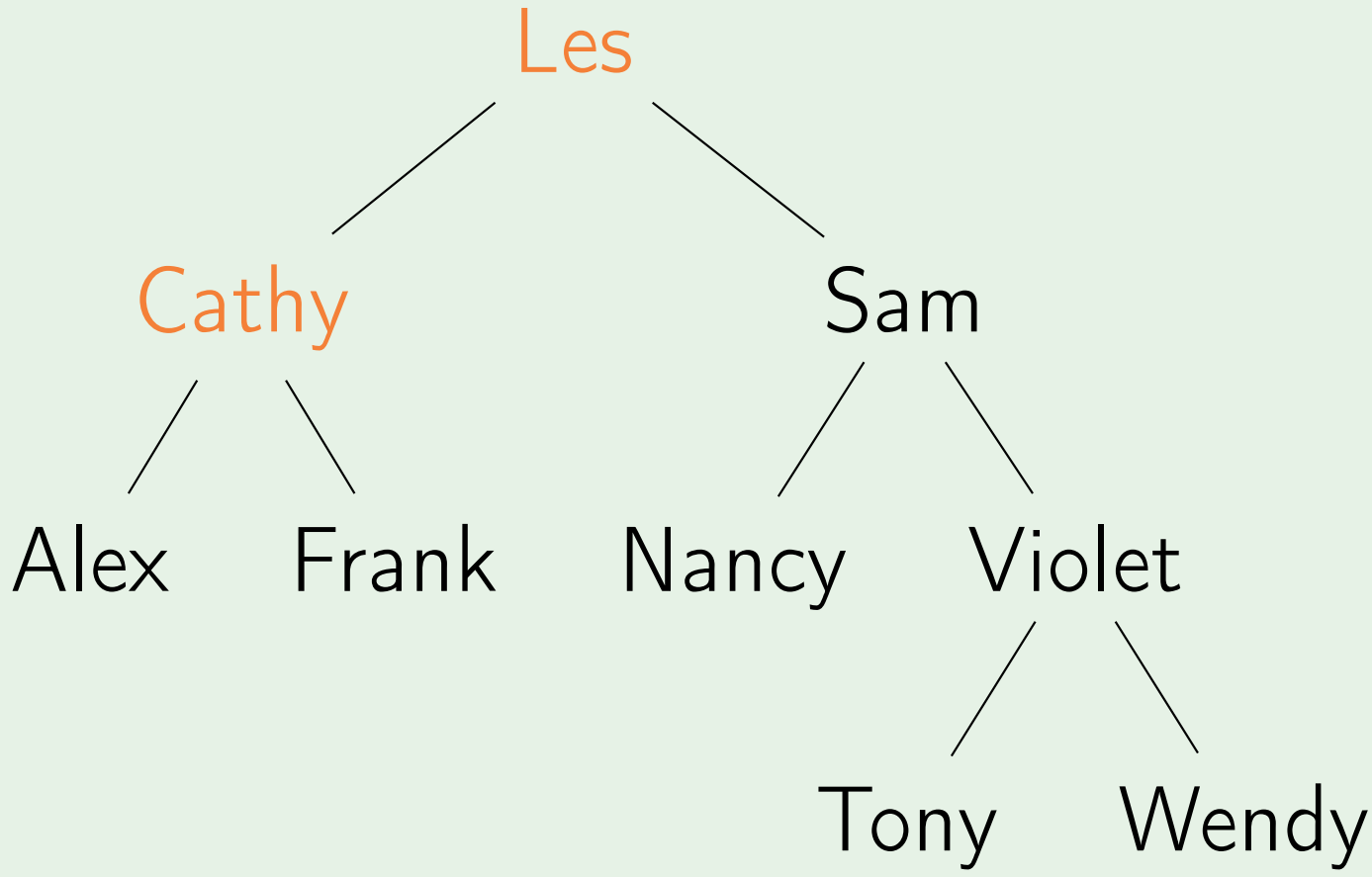
Output: Alex Frank

PostOrderTraversal



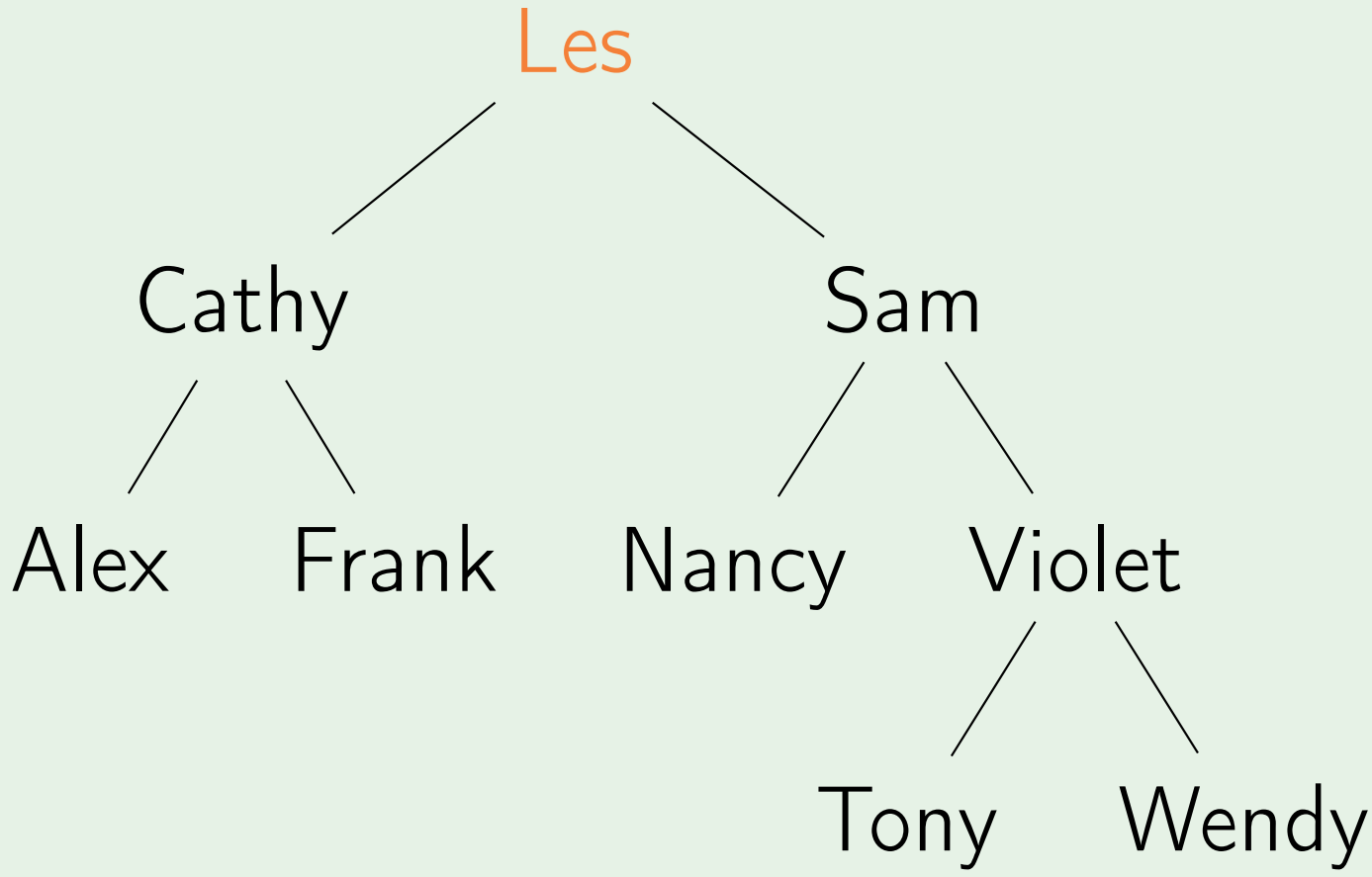
Output: Alex Frank Cathy

PostOrderTraversal



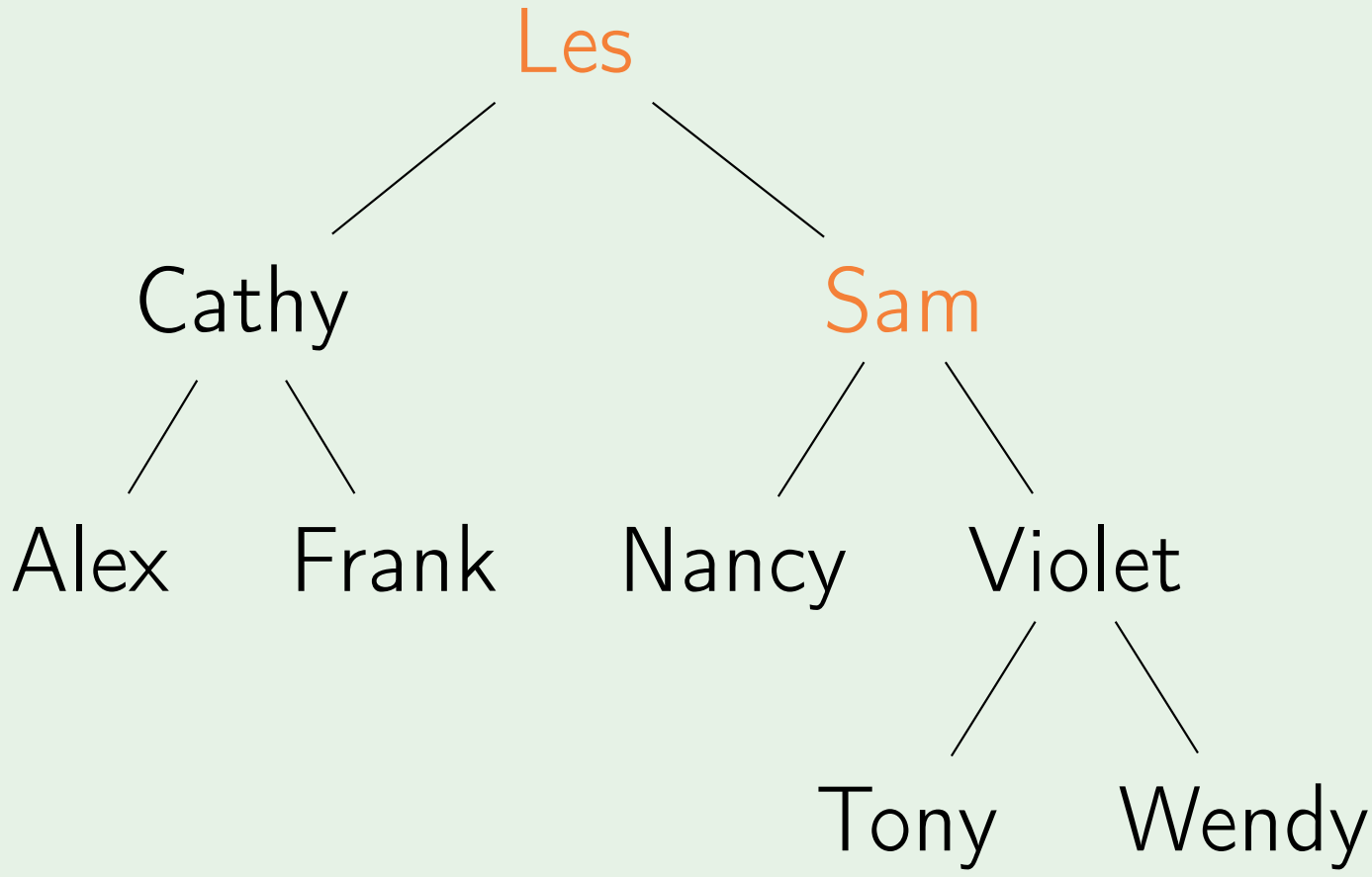
Output: Alex Frank Cathy

PostOrderTraversal



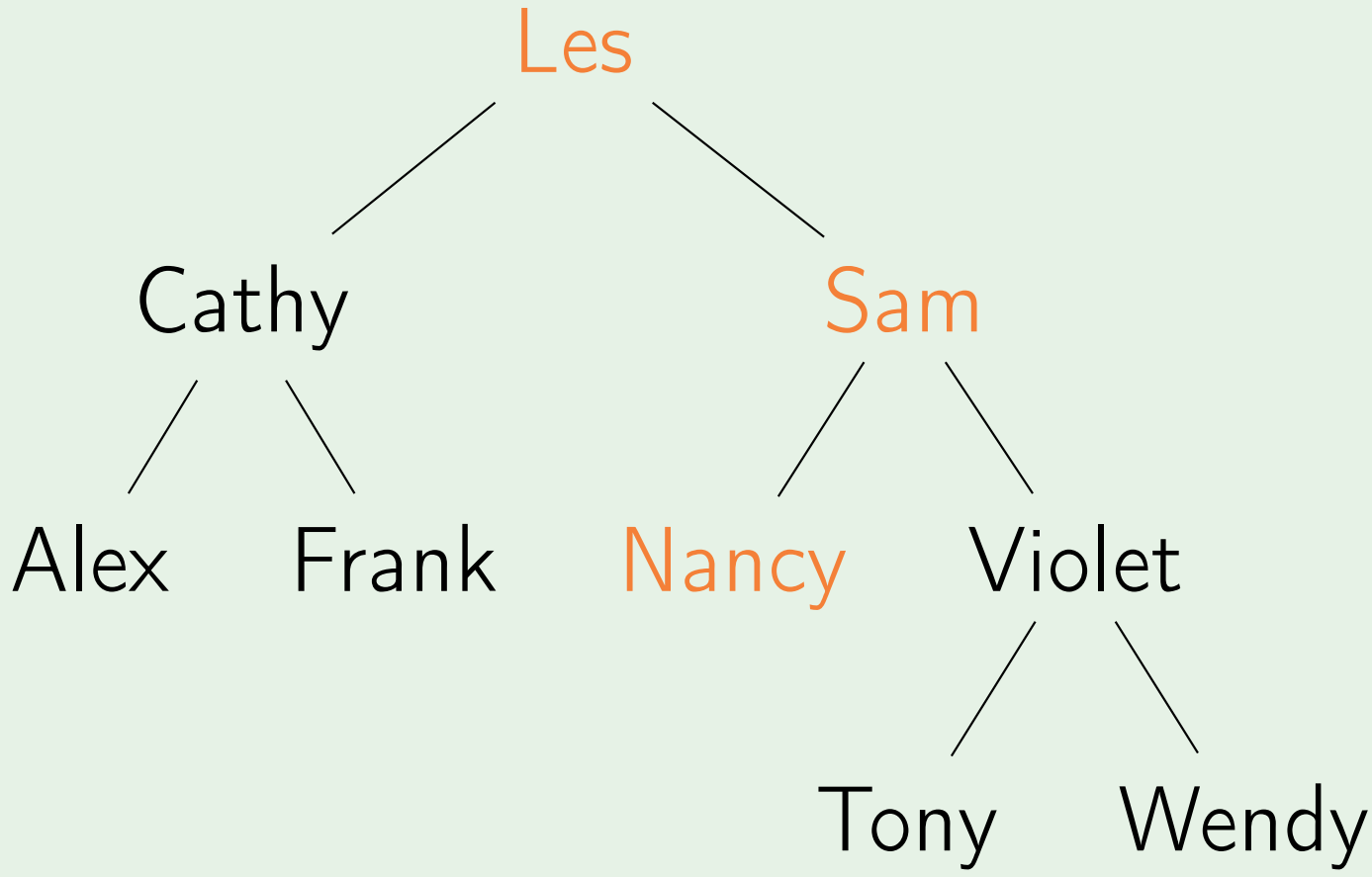
Output: Alex Frank Cathy

PostOrderTraversal



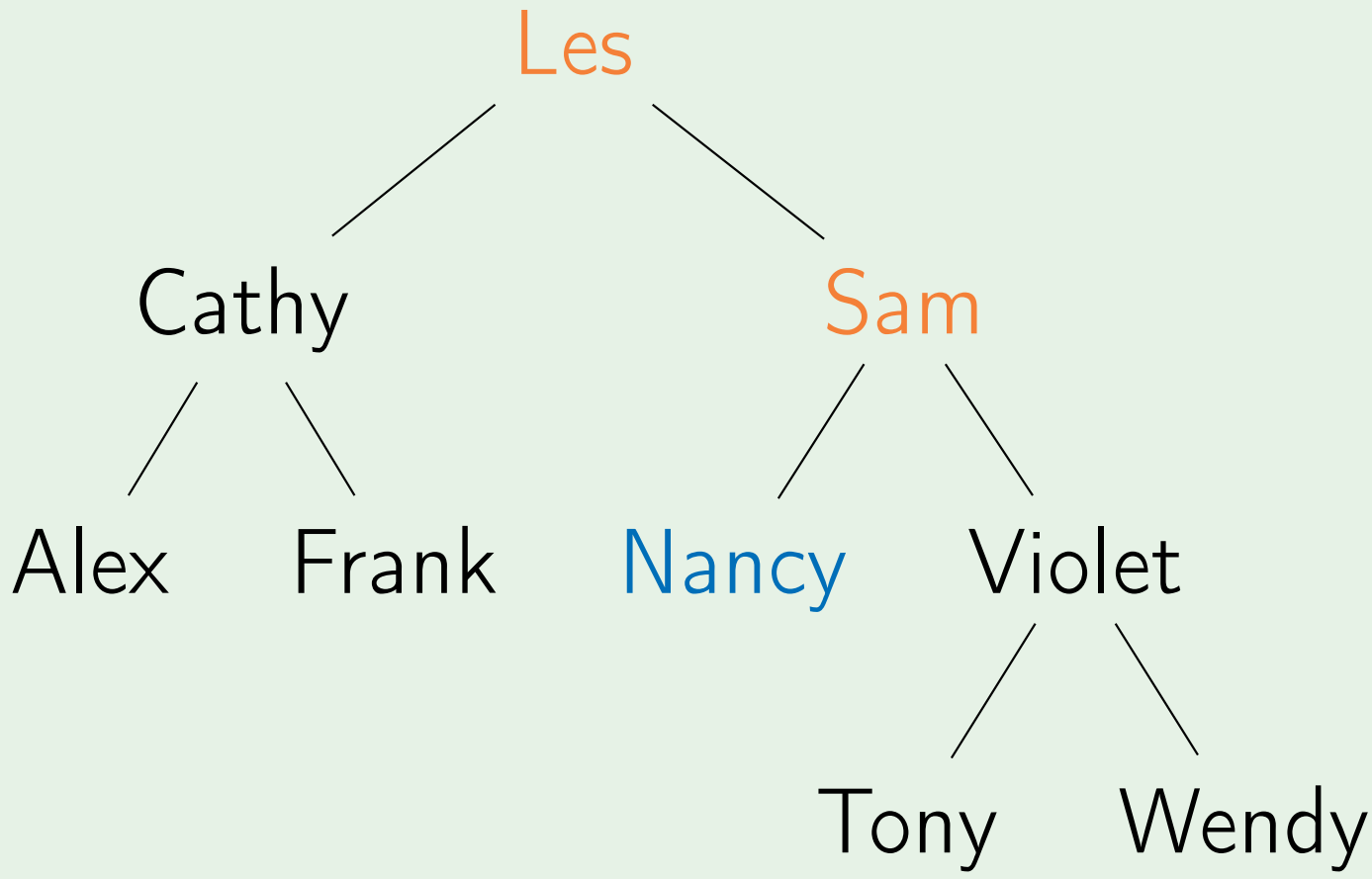
Output: Alex Frank Cathy

PostOrderTraversal



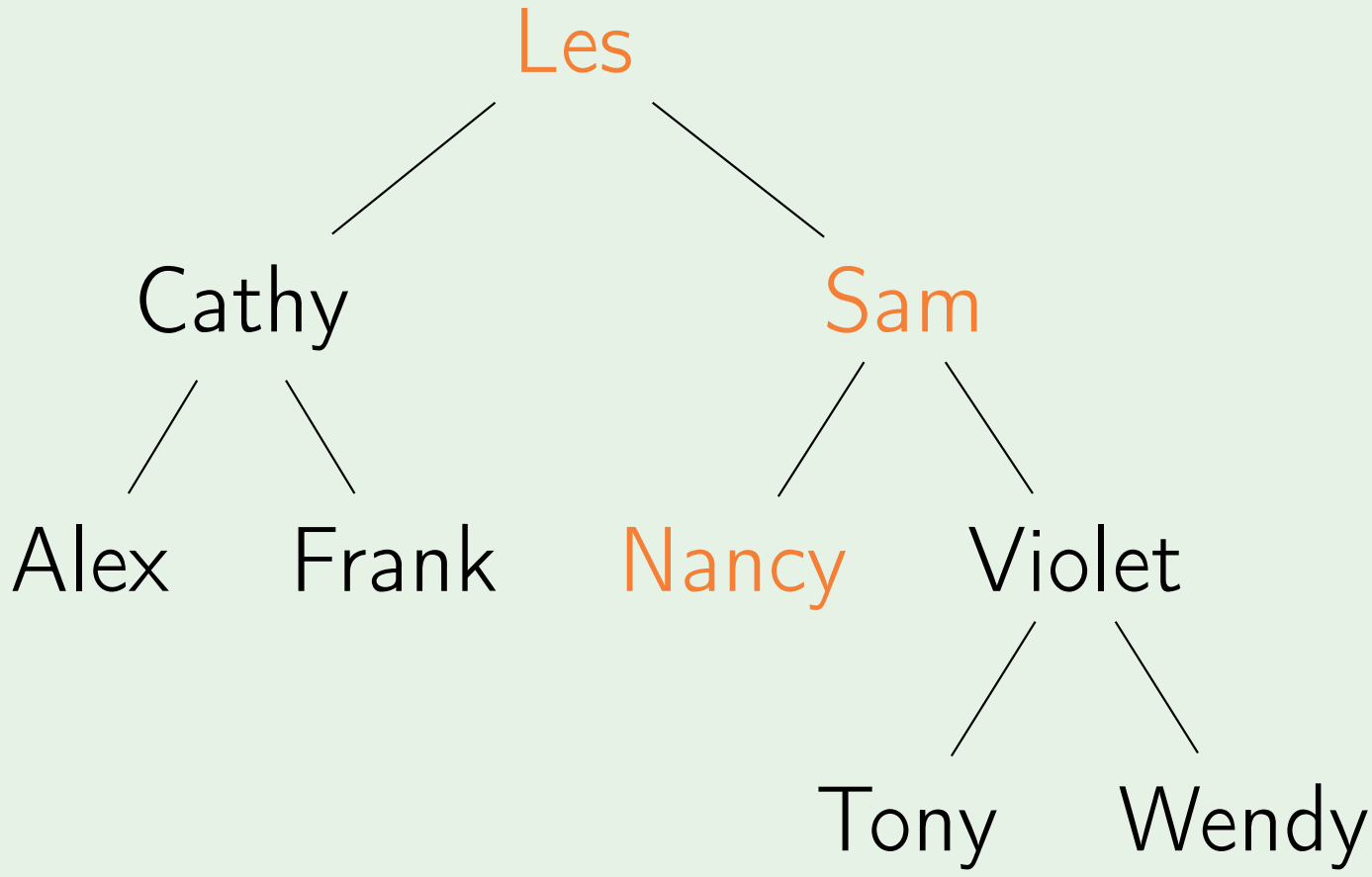
Output: Alex Frank Cathy

PostOrderTraversal



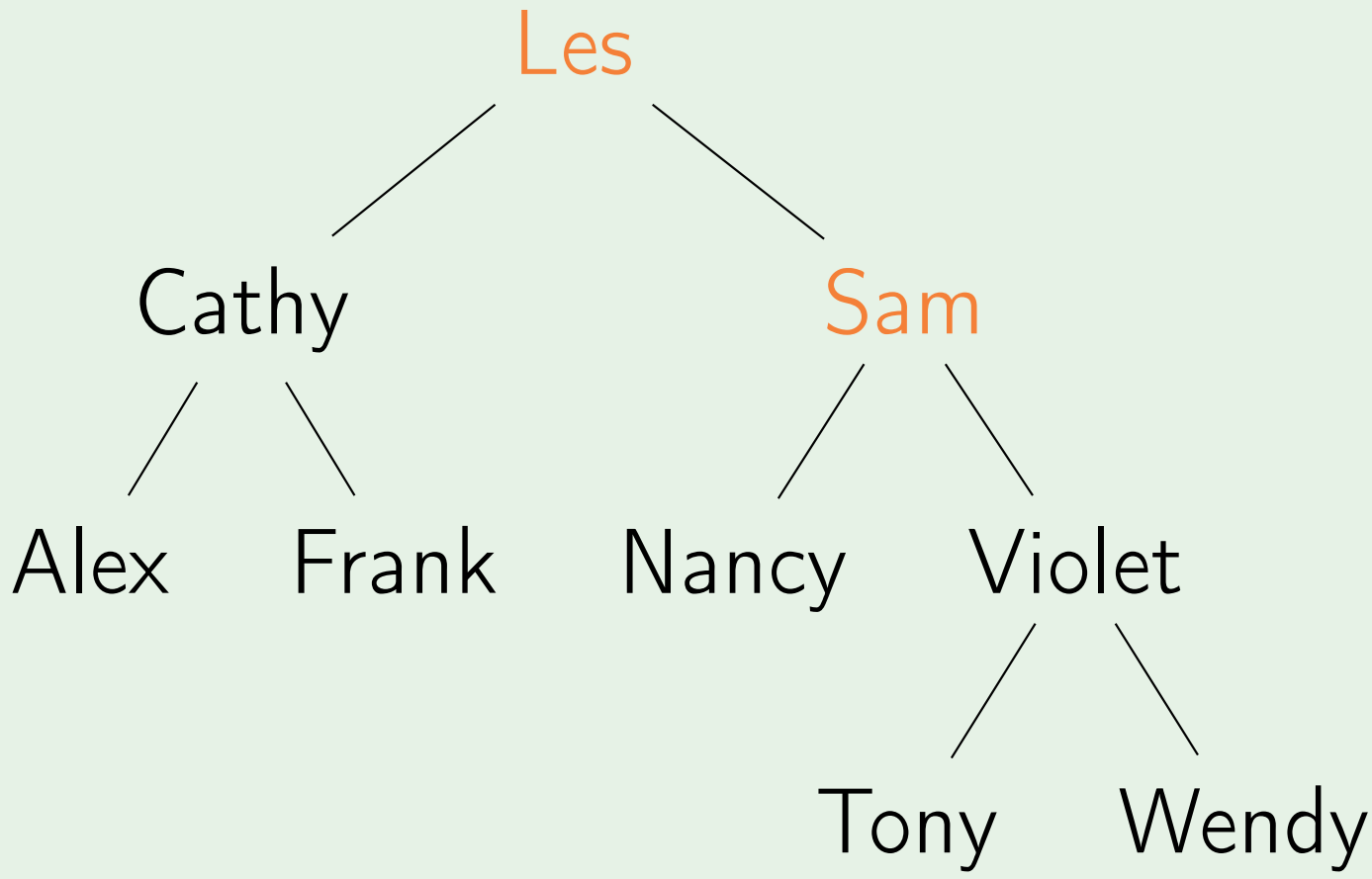
Output: Alex Frank Cathy Nancy

PostOrderTraversal



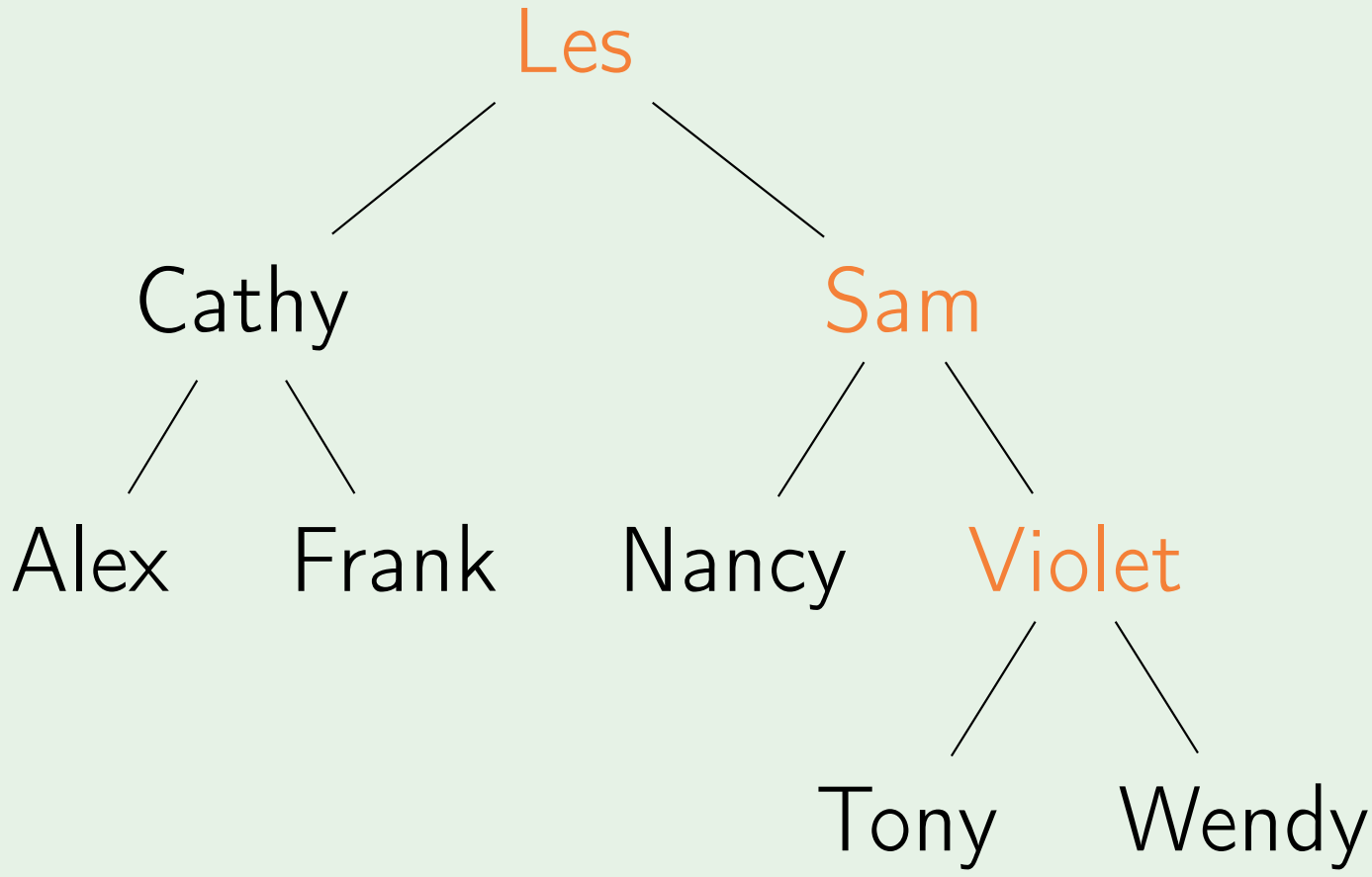
Output: Alex Frank Cathy Nancy

PostOrderTraversal



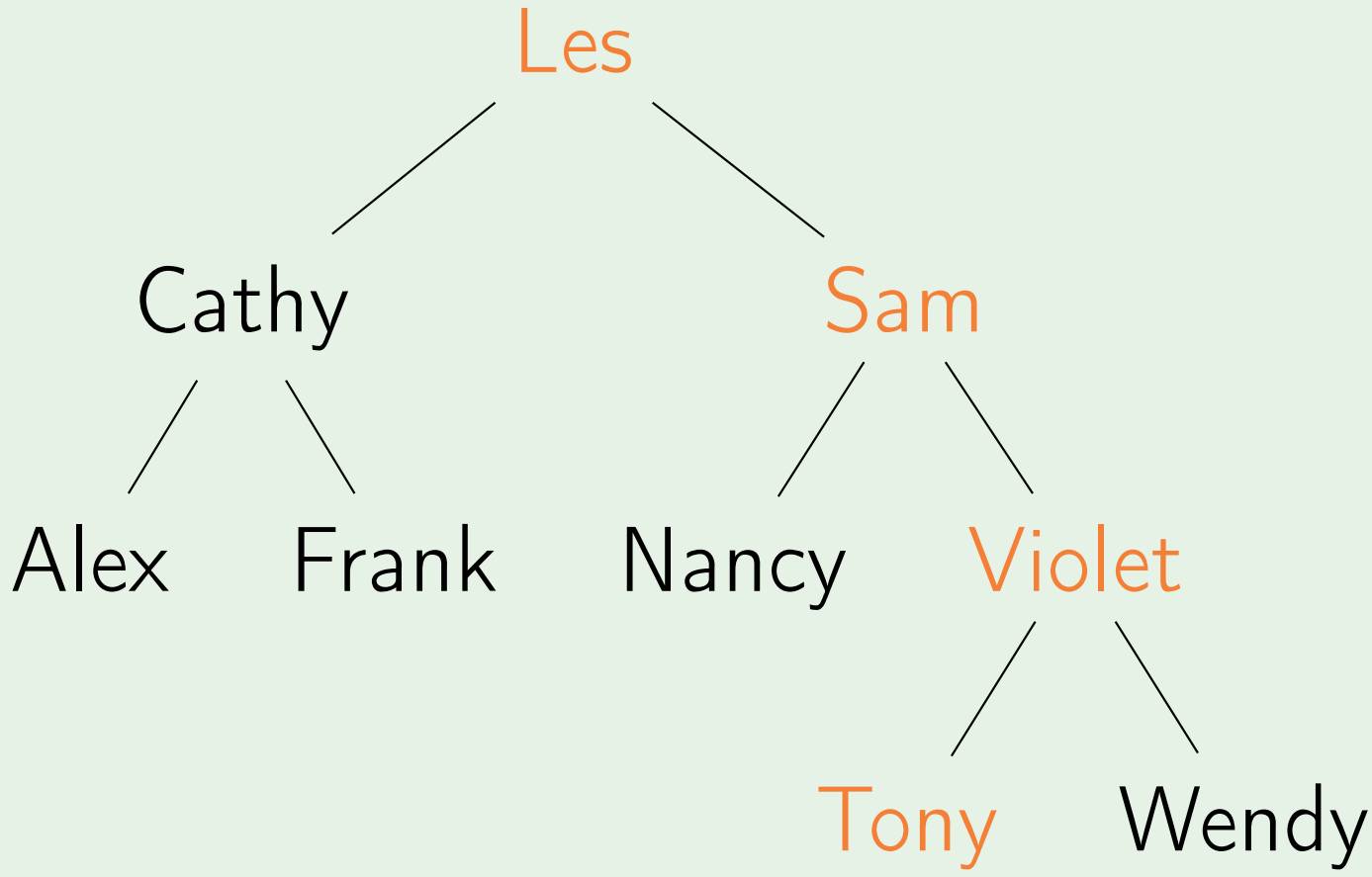
Output: Alex Frank Cathy Nancy

PostOrderTraversal



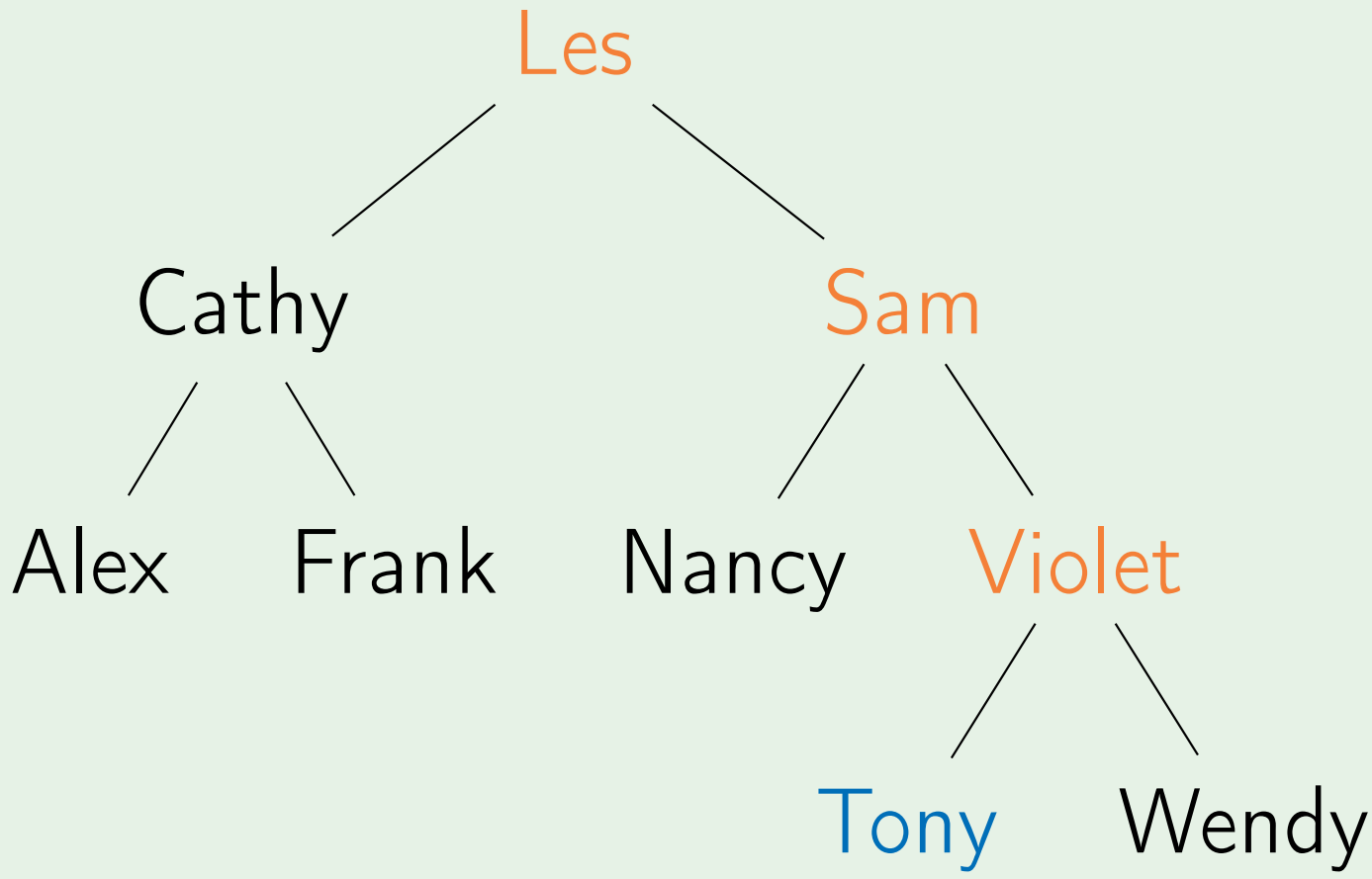
Output: Alex Frank Cathy Nancy

PostOrderTraversal



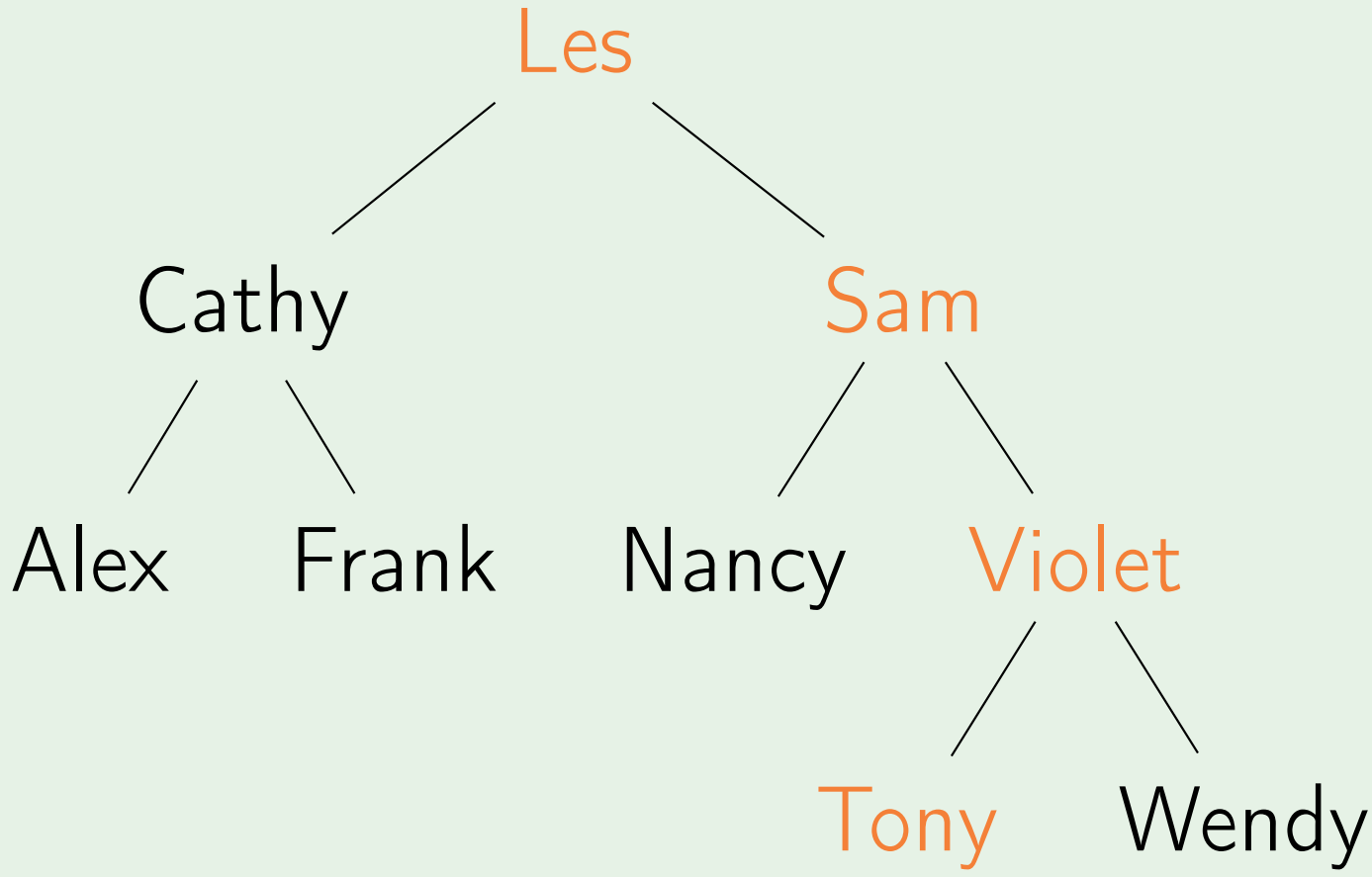
Output: Alex Frank Cathy Nancy

PostOrderTraversal



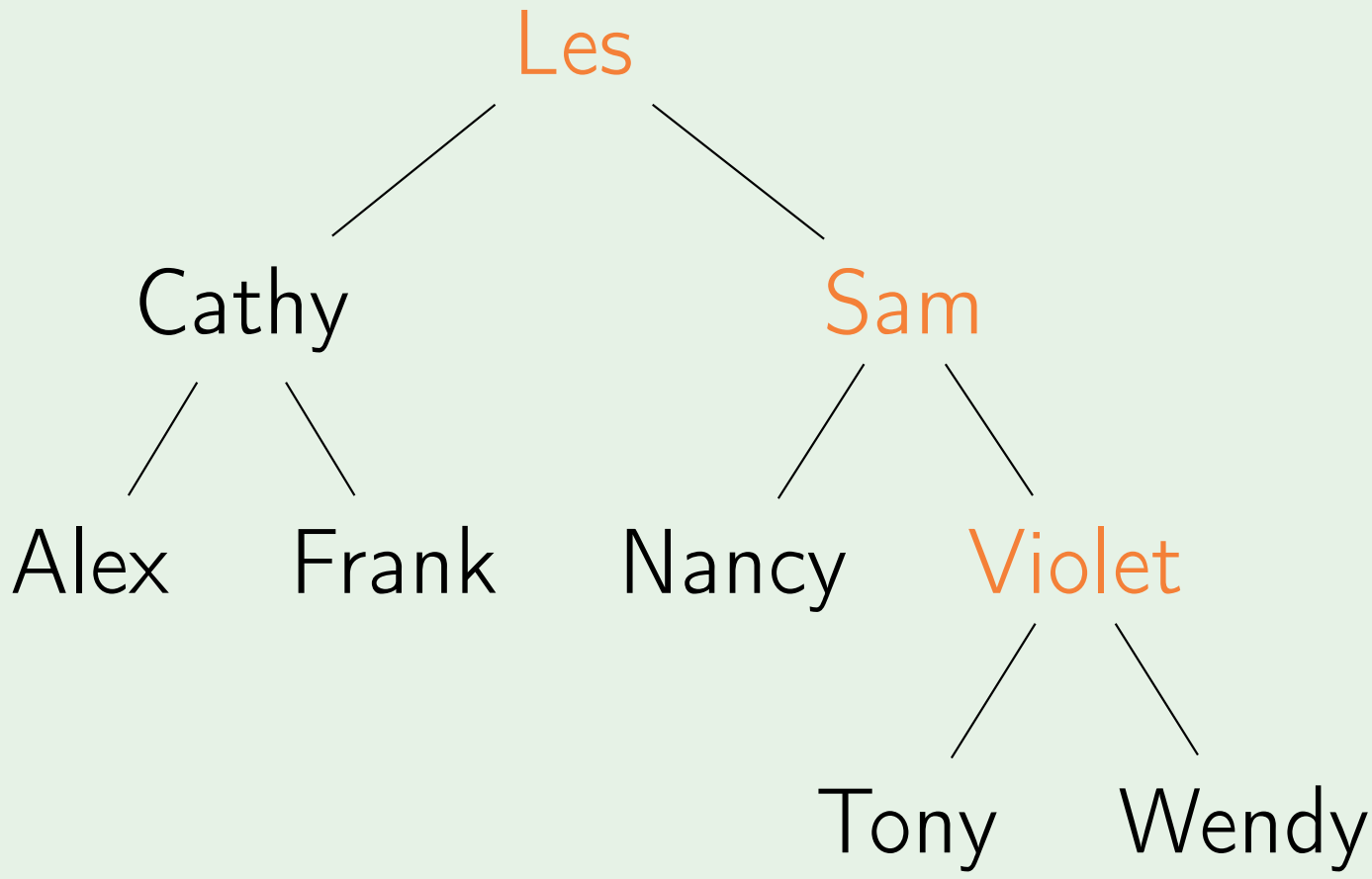
Output: Alex Frank Cathy Nancy Tony

PostOrderTraversal



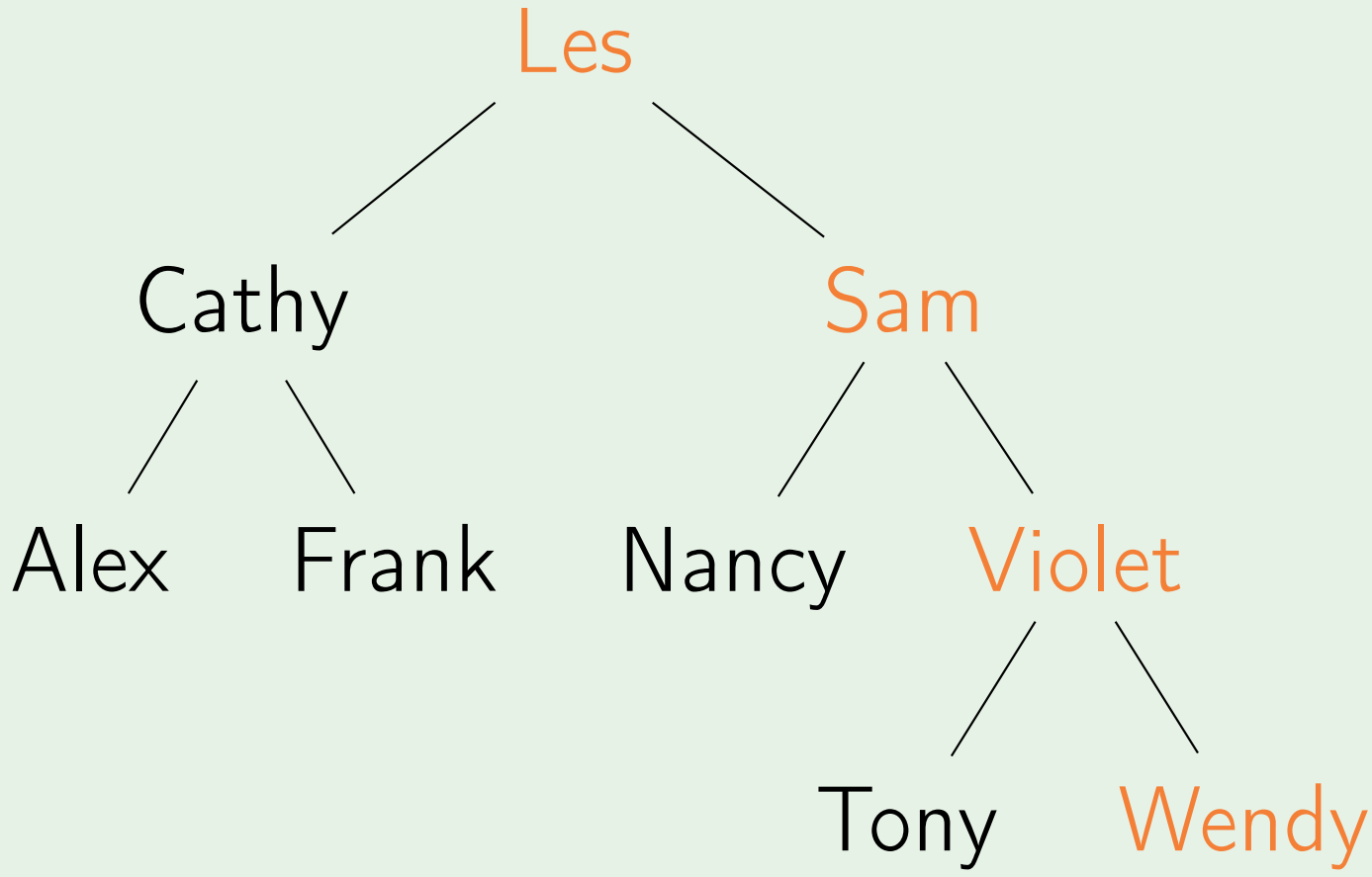
Output: Alex Frank Cathy Nancy Tony

PostOrderTraversal



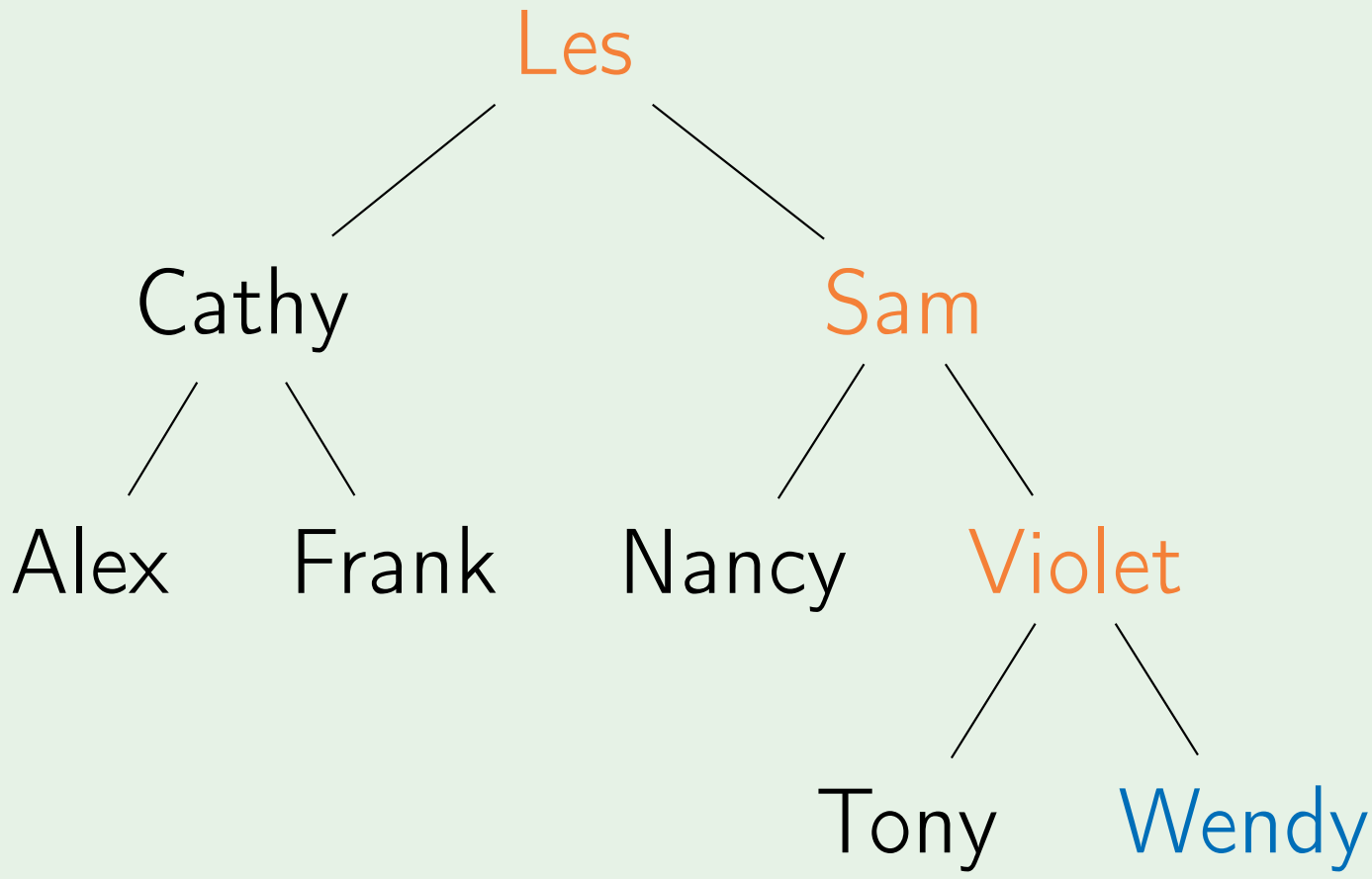
Output: Alex Frank Cathy Nancy Tony

PostOrderTraversal



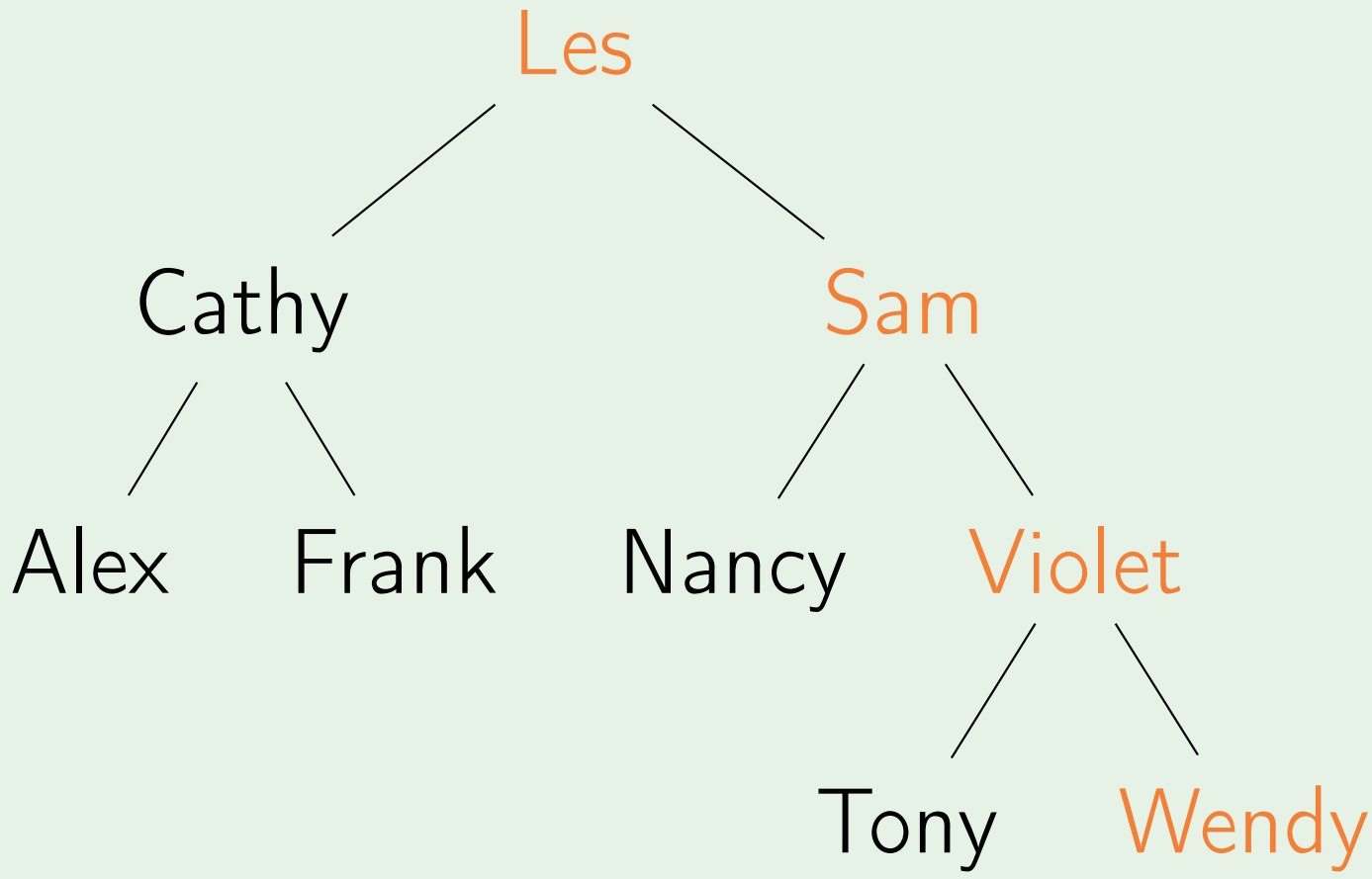
Output: Alex Frank Cathy Nancy Tony

PostOrderTraversal



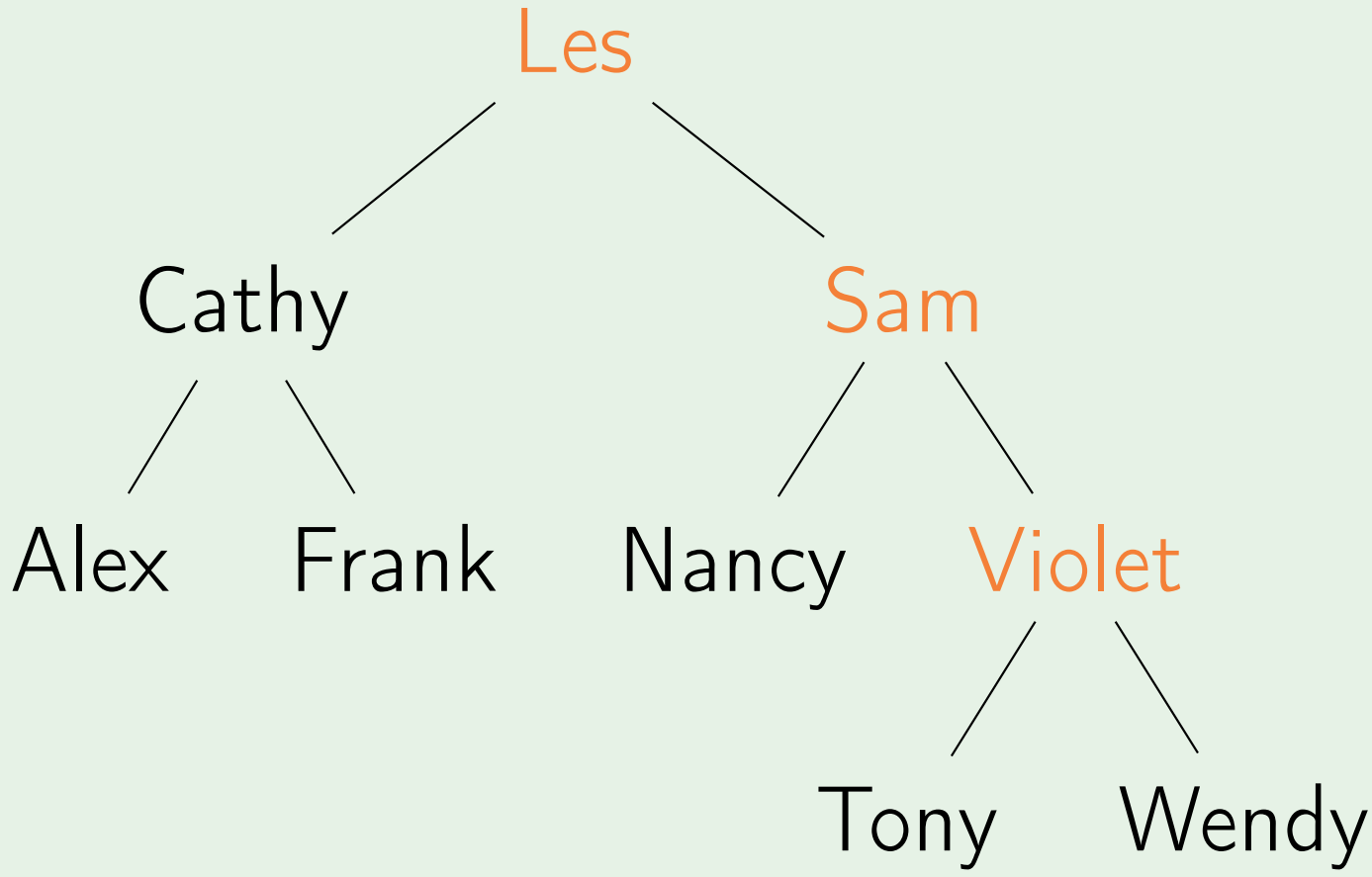
Output: Alex Frank Cathy Nancy Tony Wendy

PostOrderTraversal



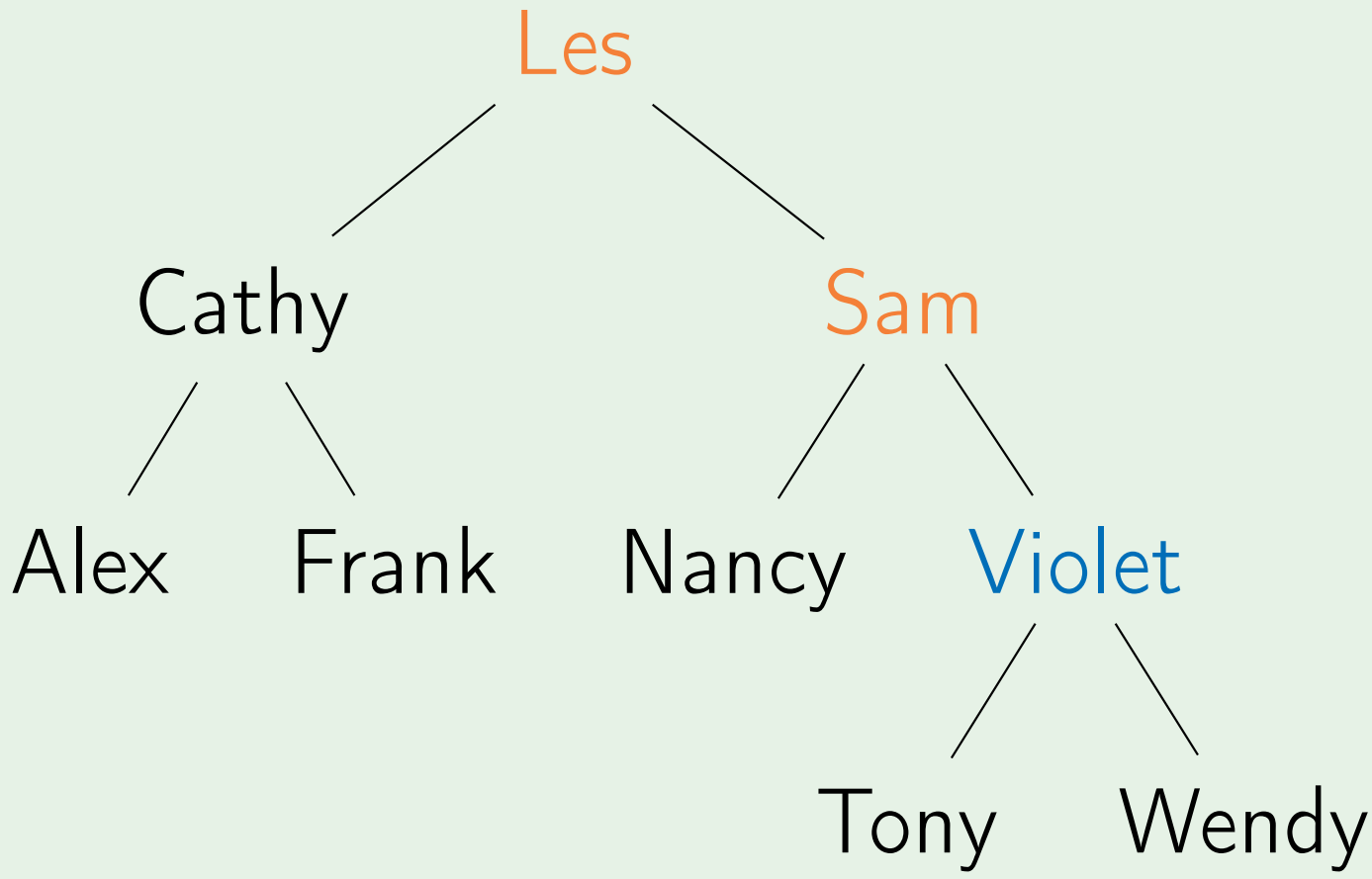
Output: Alex Frank Cathy Nancy Tony Wendy

PostOrderTraversal



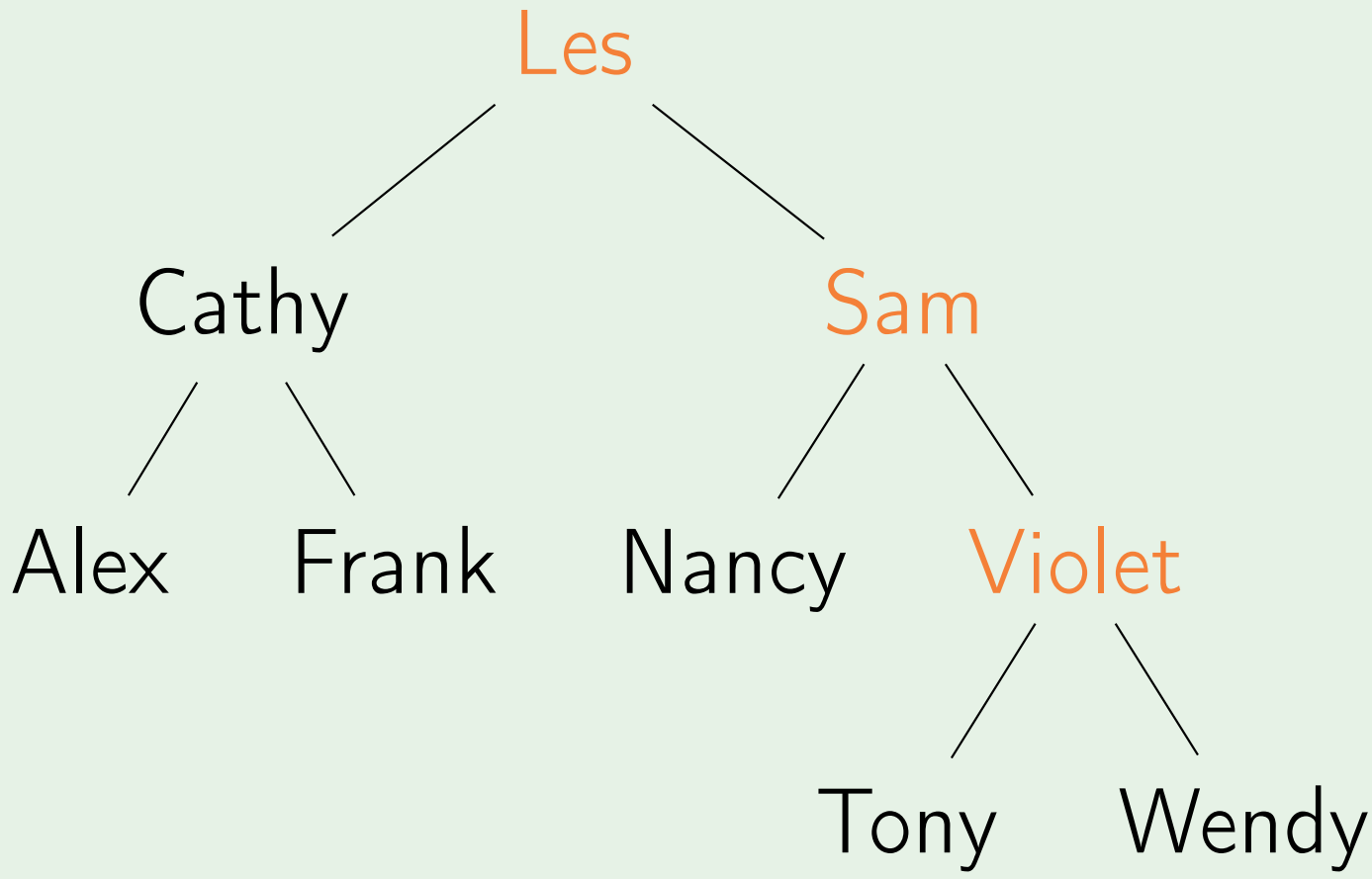
Output: Alex Frank Cathy Nancy Tony Wendy

PostOrderTraversal



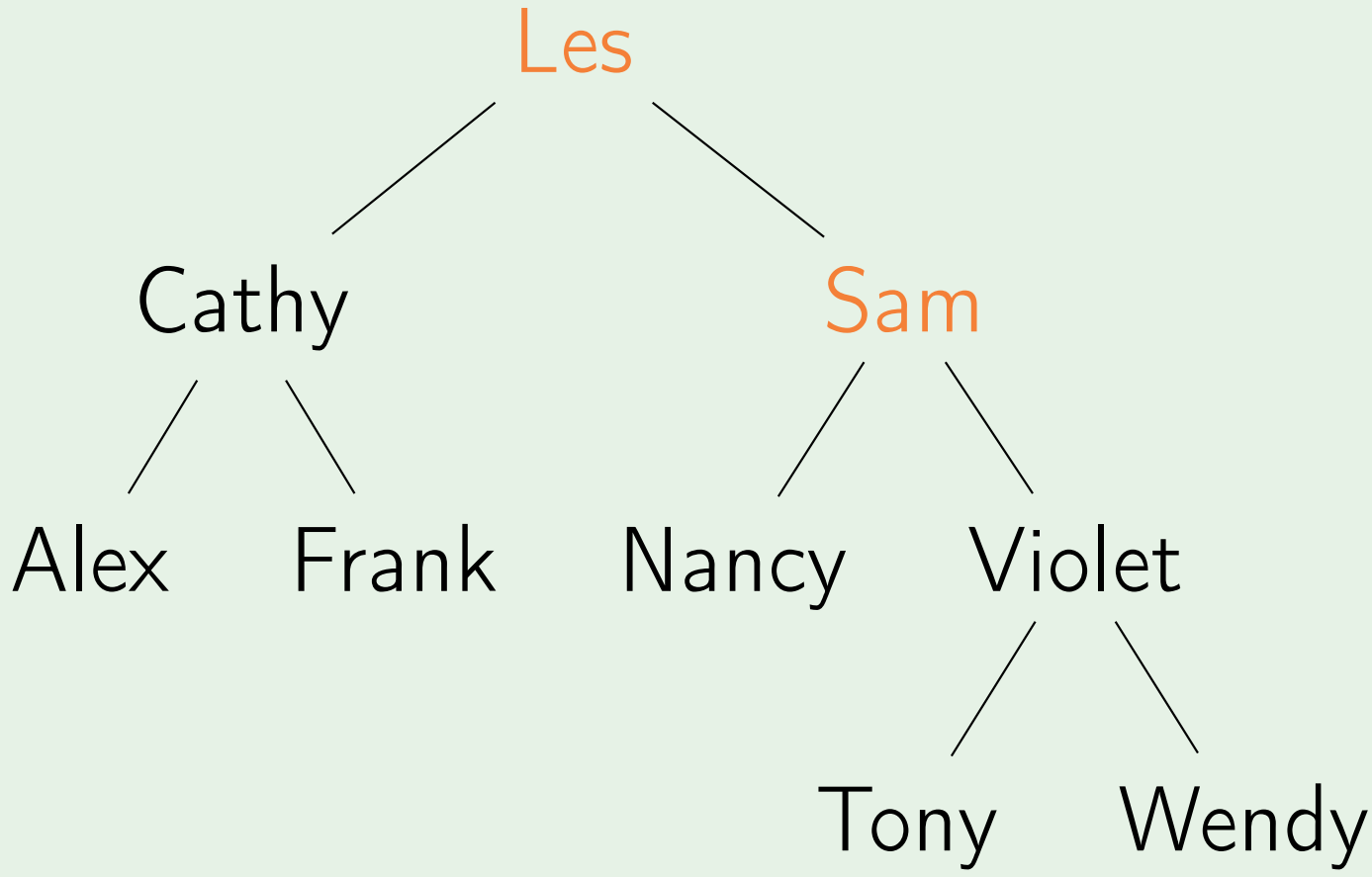
Output: Alex Frank Cathy Nancy Tony Wendy Violet

PostOrderTraversal



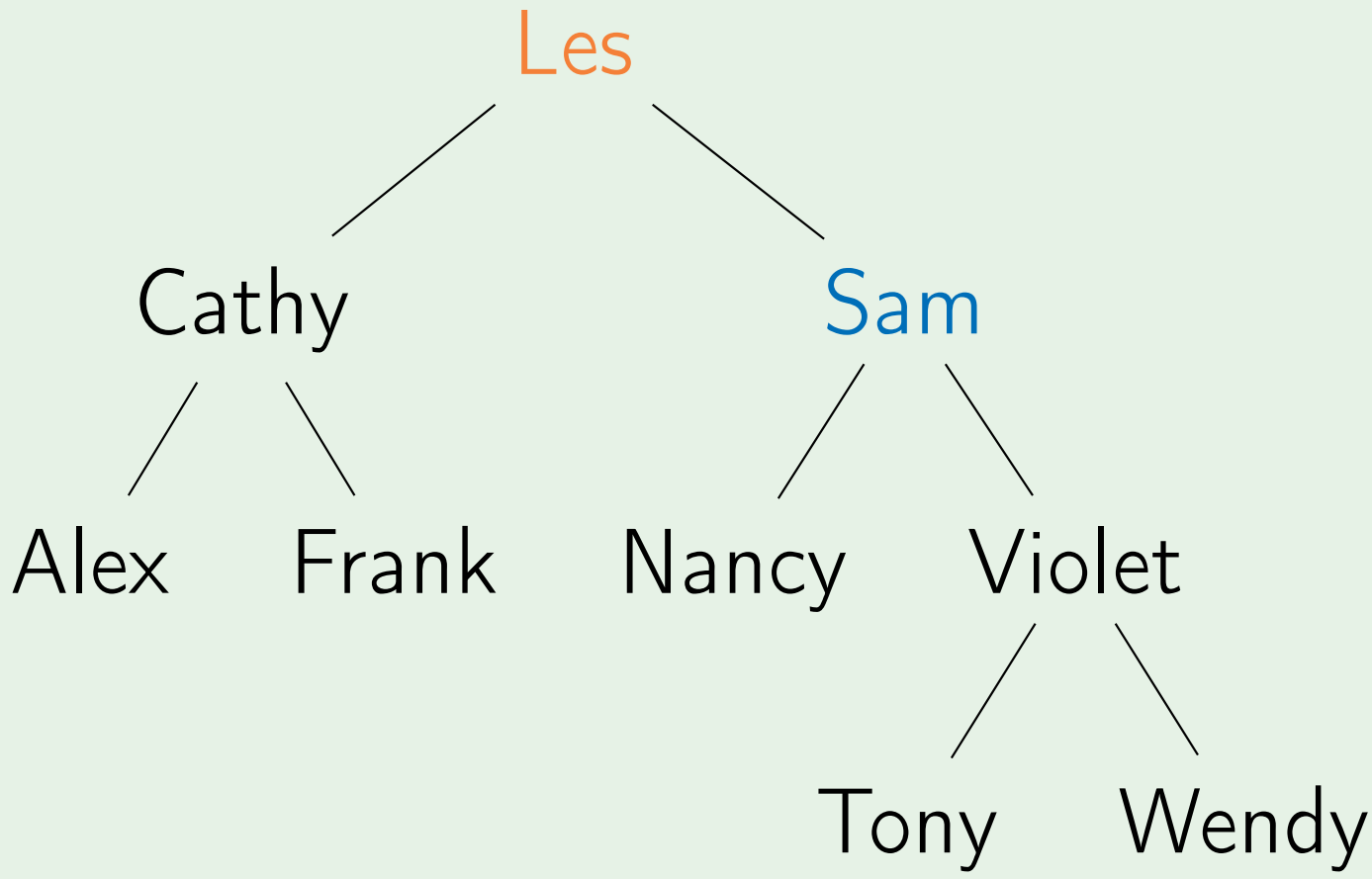
Output: Alex Frank Cathy Nancy Tony Wendy Violet

PostOrderTraversal



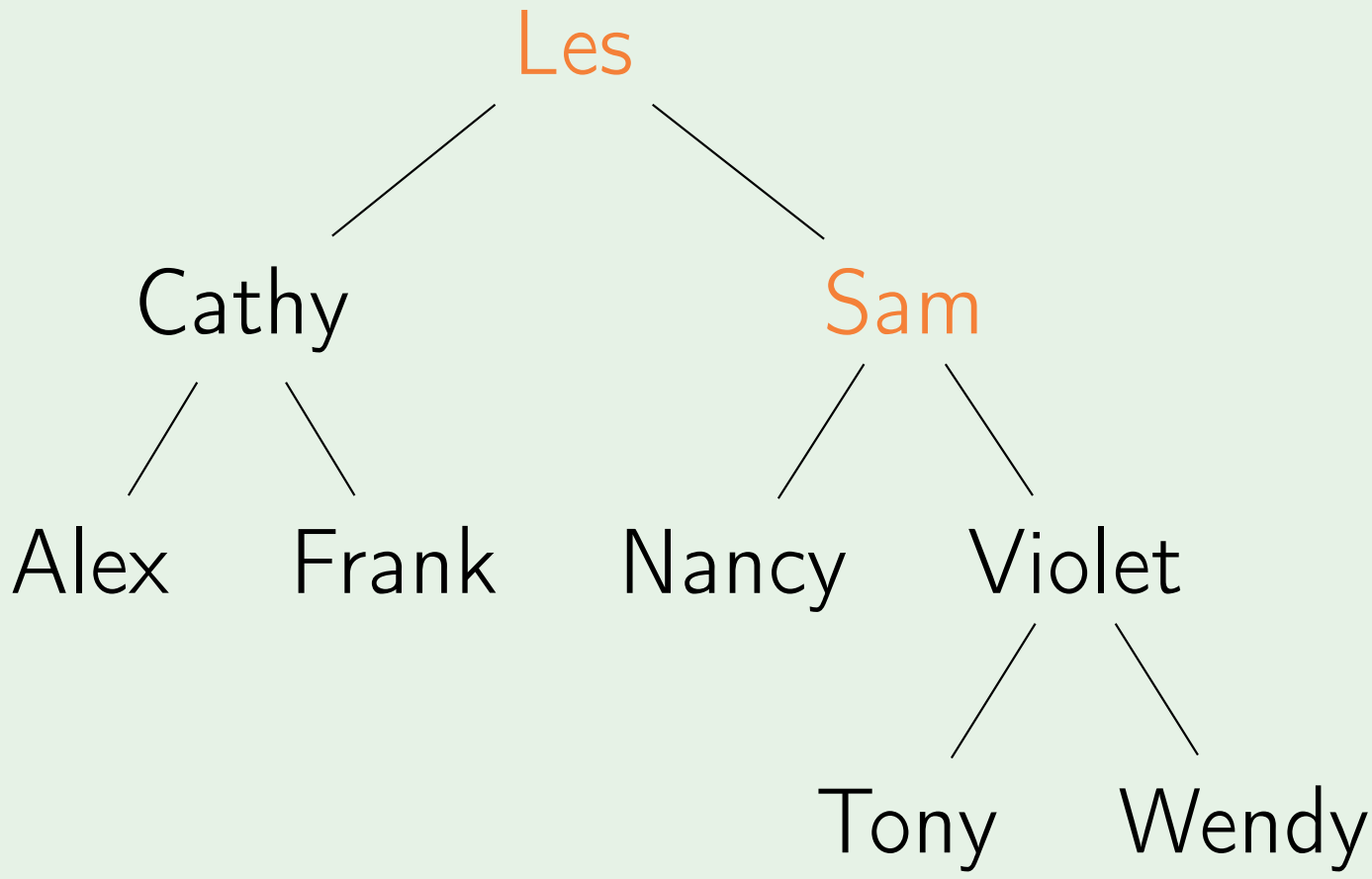
Output: Alex Frank Cathy Nancy Tony Wendy Violet

PostOrderTraversal



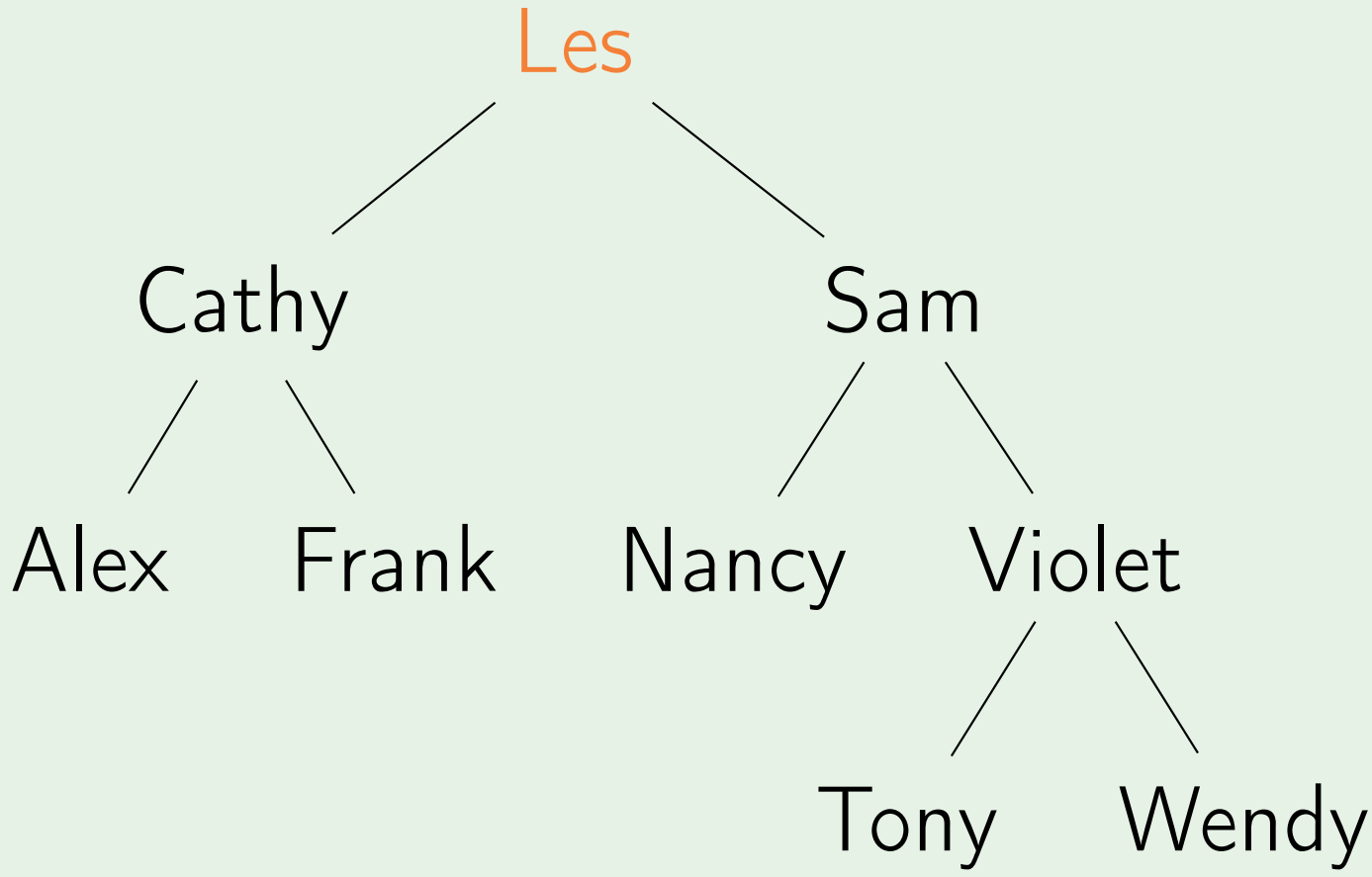
Output: Alex Frank Cathy Nancy Tony Wendy Violet Sam

PostOrderTraversal



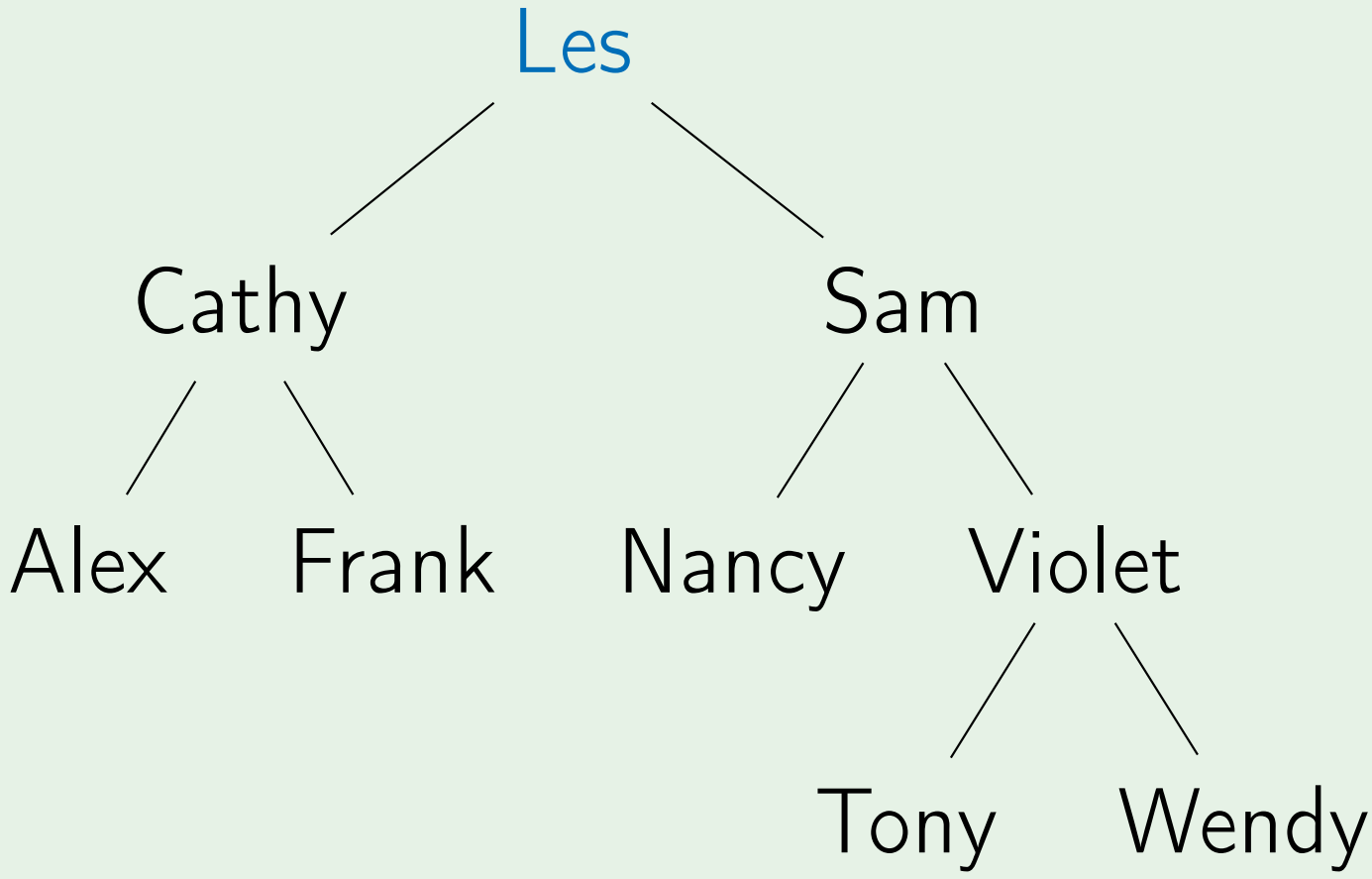
Output: Alex Frank Cathy Nancy Tony Wendy Violet Sam

PostOrderTraversal



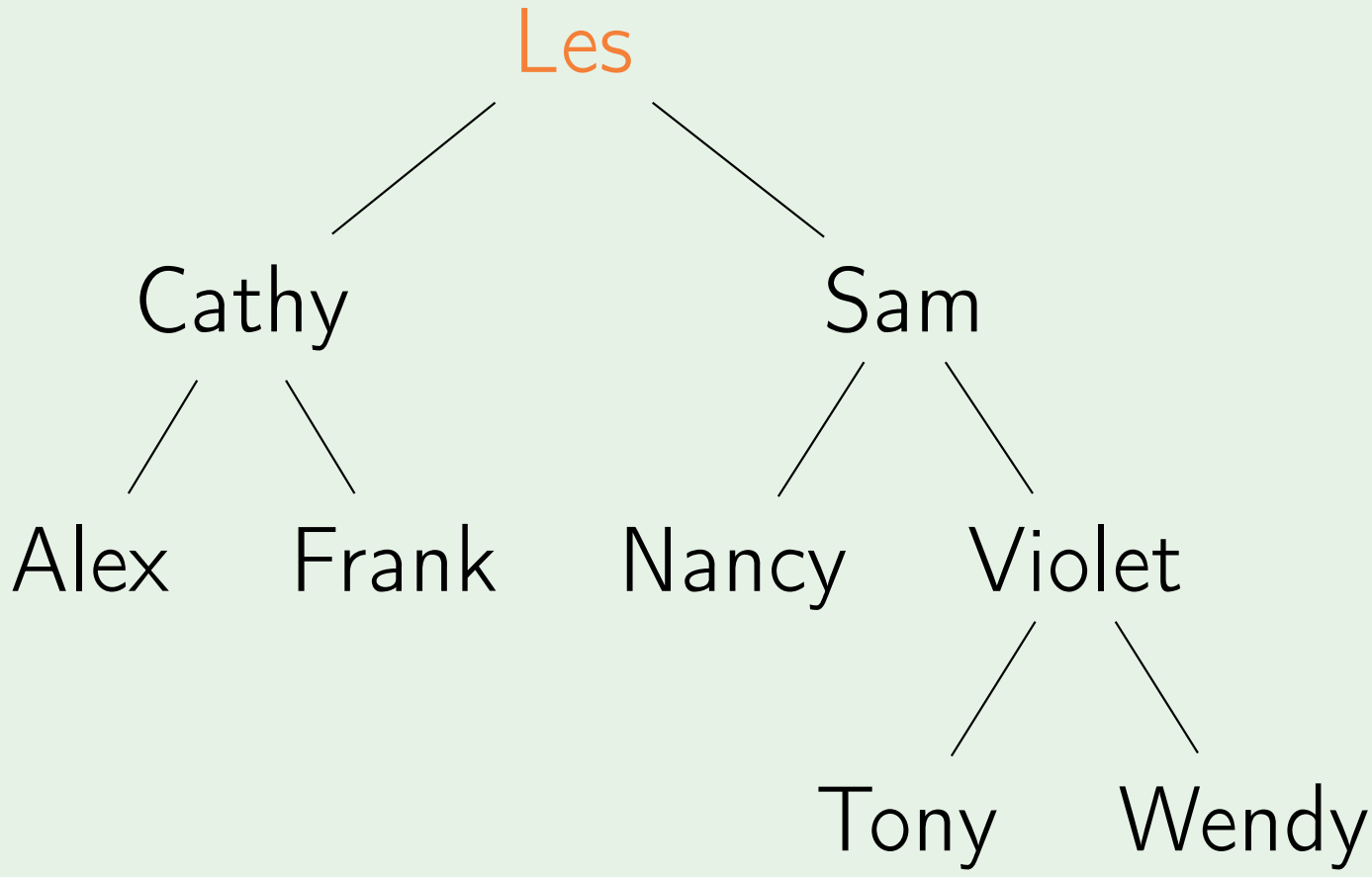
Output: Alex Frank Cathy Nancy Tony Wendy Violet Sam

PostOrderTraversal



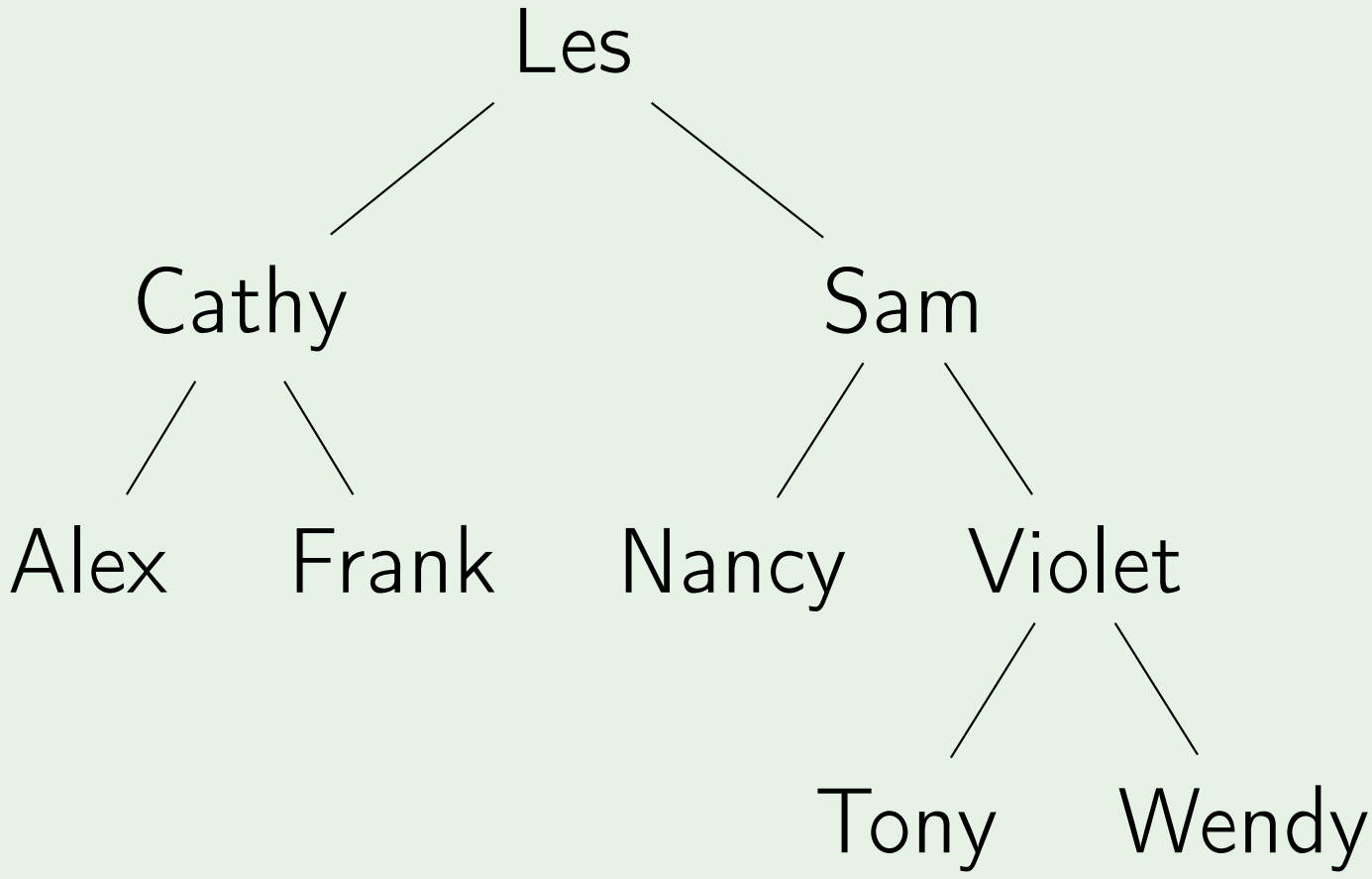
Output: Alex Frank Cathy Nancy Tony Wendy Violet Sam Les

PostOrderTraversal



Output: Alex Frank Cathy Nancy Tony Wendy Violet Sam Les

PostOrderTraversal



Output: Alex Frank Cathy Nancy Tony Wendy Violet Sam Les

Breadth-first

```
LevelTraversal(tree)
if tree = nil: return

Queue q
q.Enqueue(tree)
```

Breadth-first

```
LevelTraversal(tree)
if tree = nil:  return

Queue q
q.Enqueue(tree)
while not q.Empty():
    node ← q.Dequeue()
```

Breadth-first

```
LevelTraversal(tree)
if tree = nil:  return

Queue q
q.Enqueue(tree)
while not q.Empty():
    node ← q.Dequeue()
    Print(node)
```

Breadth-first

```
LevelTraversal(tree)
if tree = nil:  return

Queue q
q.Enqueue(tree)
while not q.Empty():
    node ← q.Dequeue()
    Print(node)
    if node.left ≠ nil:
        q.Enqueue(node.left)
```

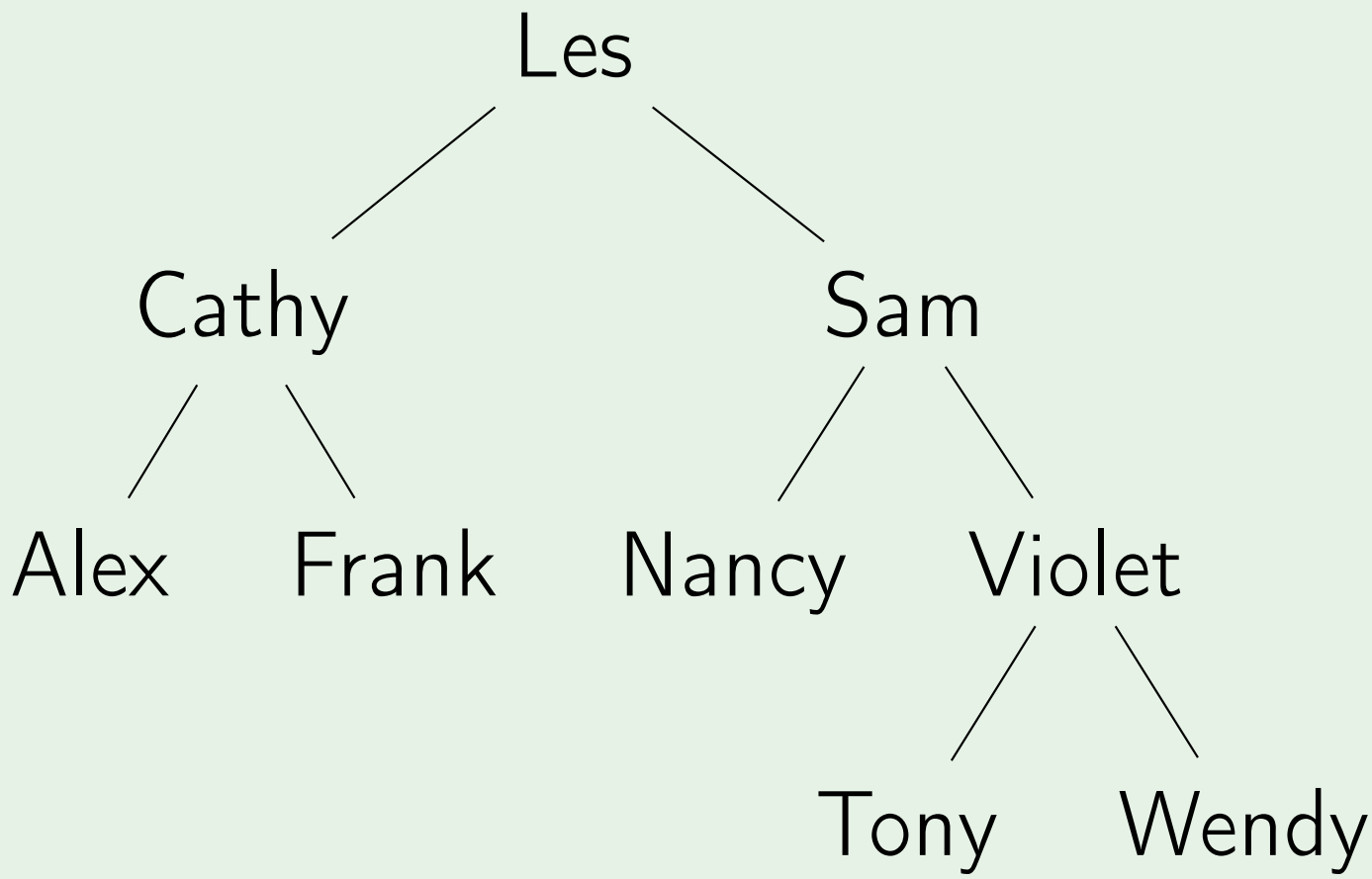
Breadth-first

LevelTraversal(*tree*)

```
if tree = nil :   return

Queue q
q.Enqueue(tree)
while not q.Empty() :
    node ← q.Dequeue()
    Print(node)
    if node.left ≠ nil :
        q.Enqueue(node.left)
    if node.right ≠ nil :
        q.Enqueue(node.right)
```

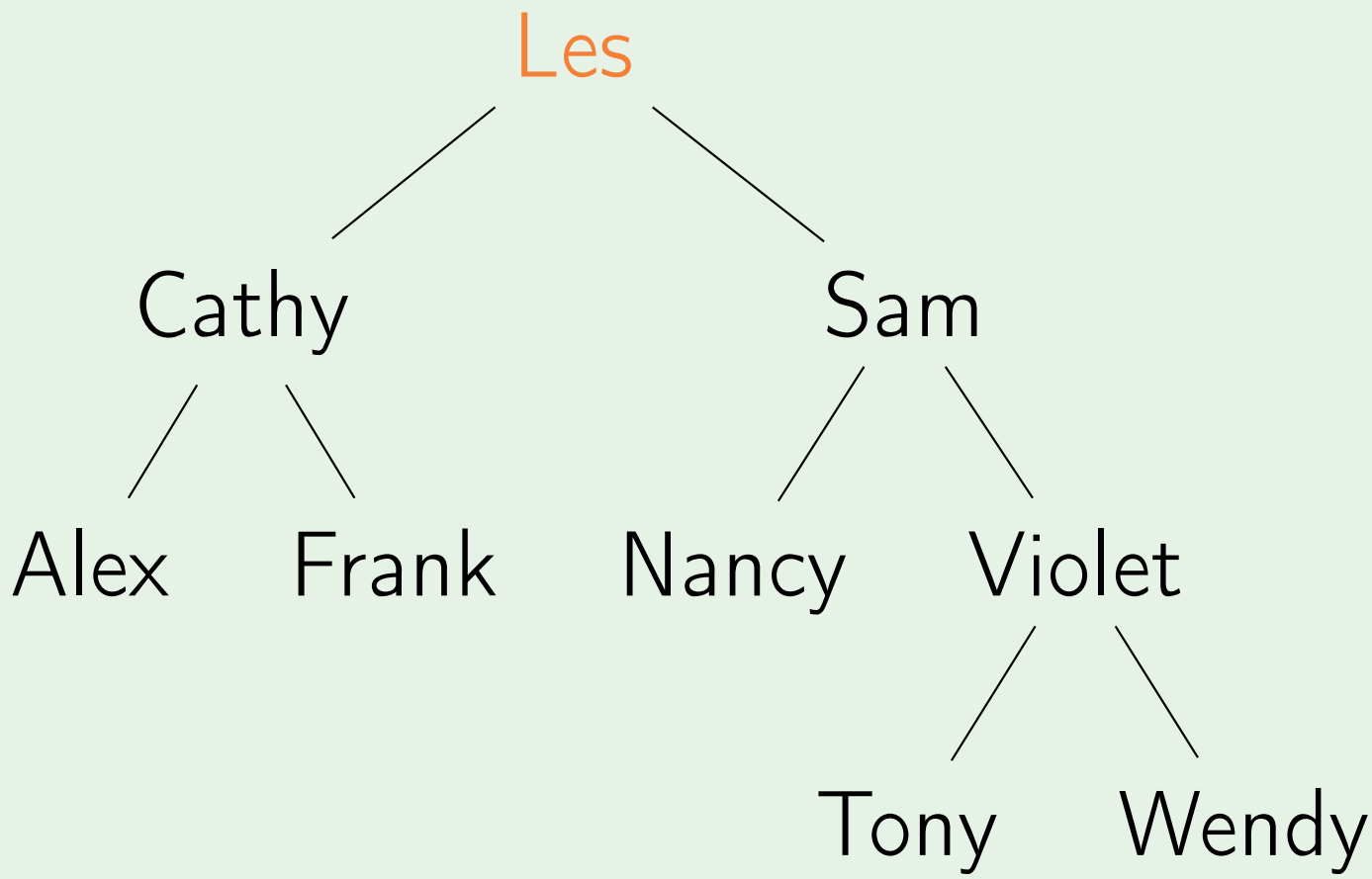
LevelTraversal



Output:

Queue: Les

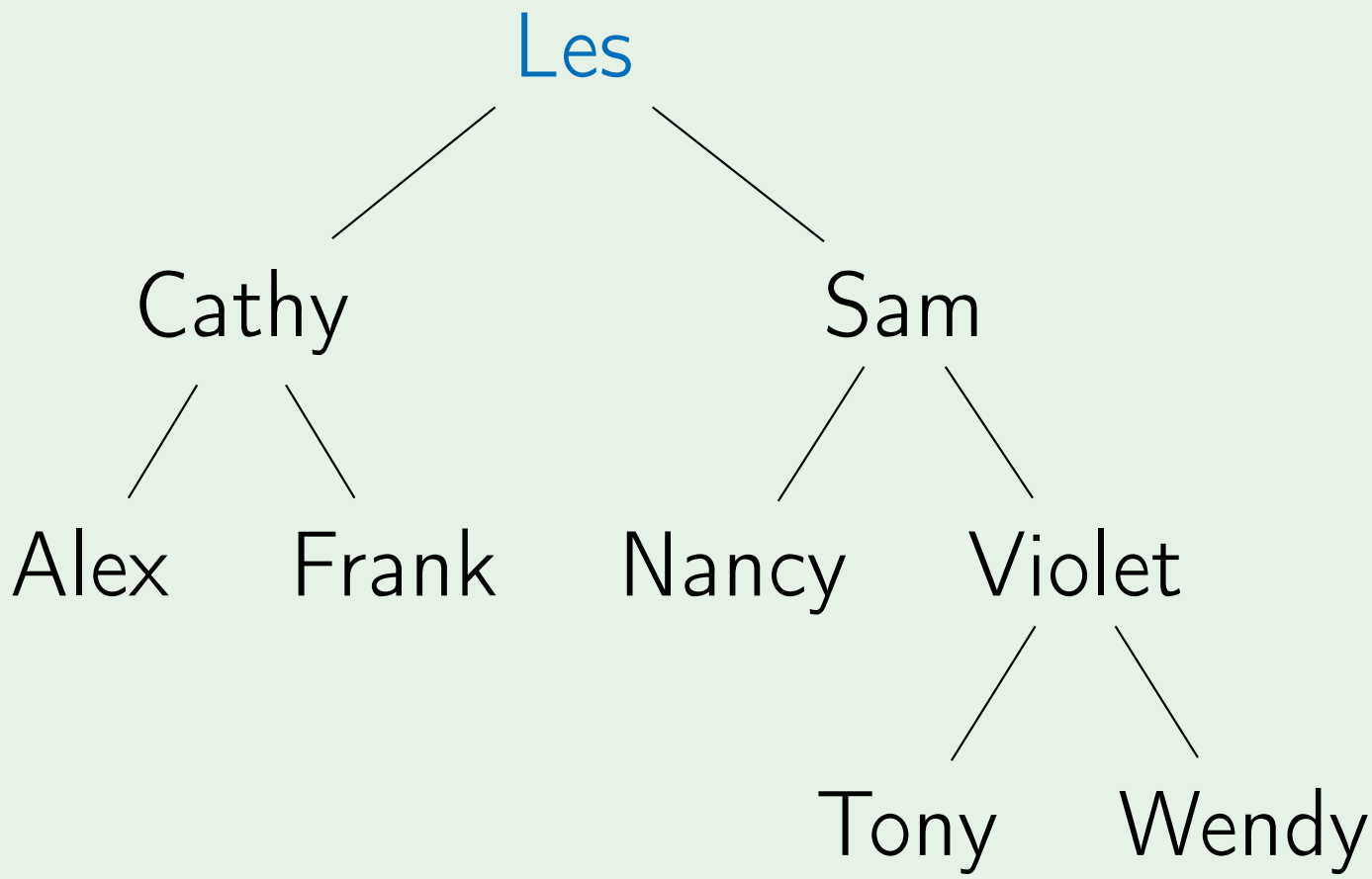
LevelTraversal



Output:

Queue:

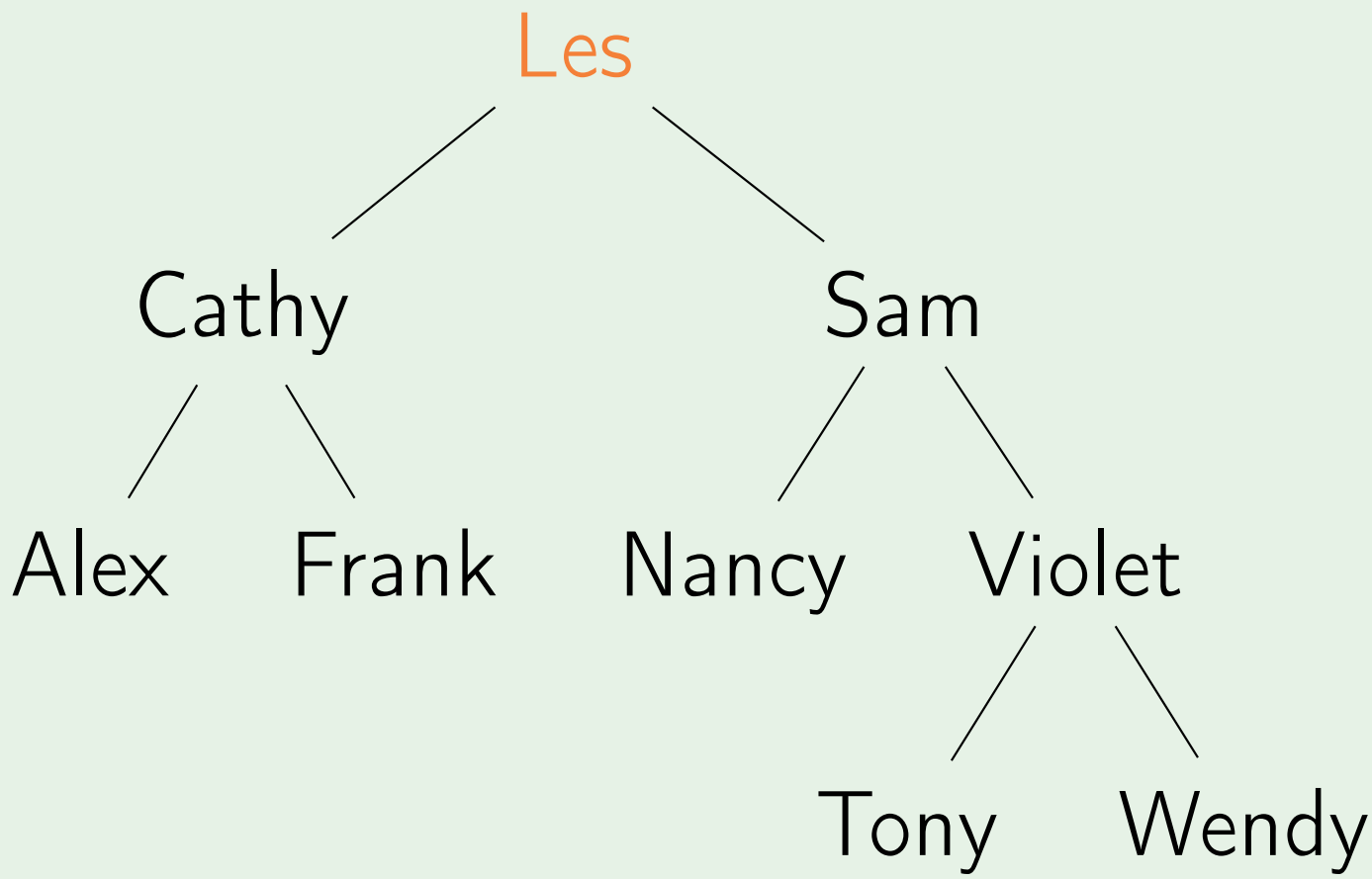
LevelTraversal



Output: Les

Queue:

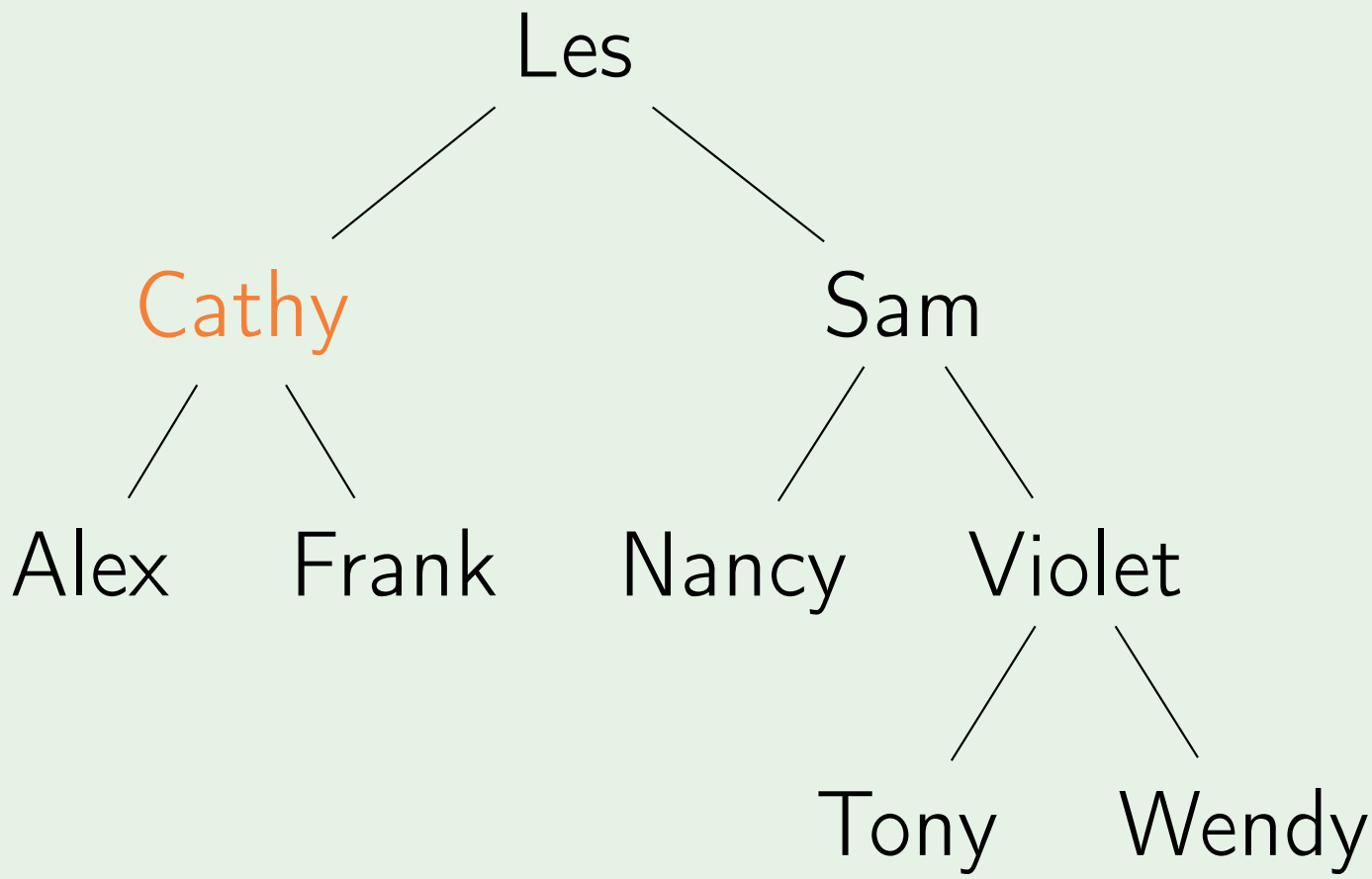
LevelTraversal



Output: Les

Queue: Cathy, Sam

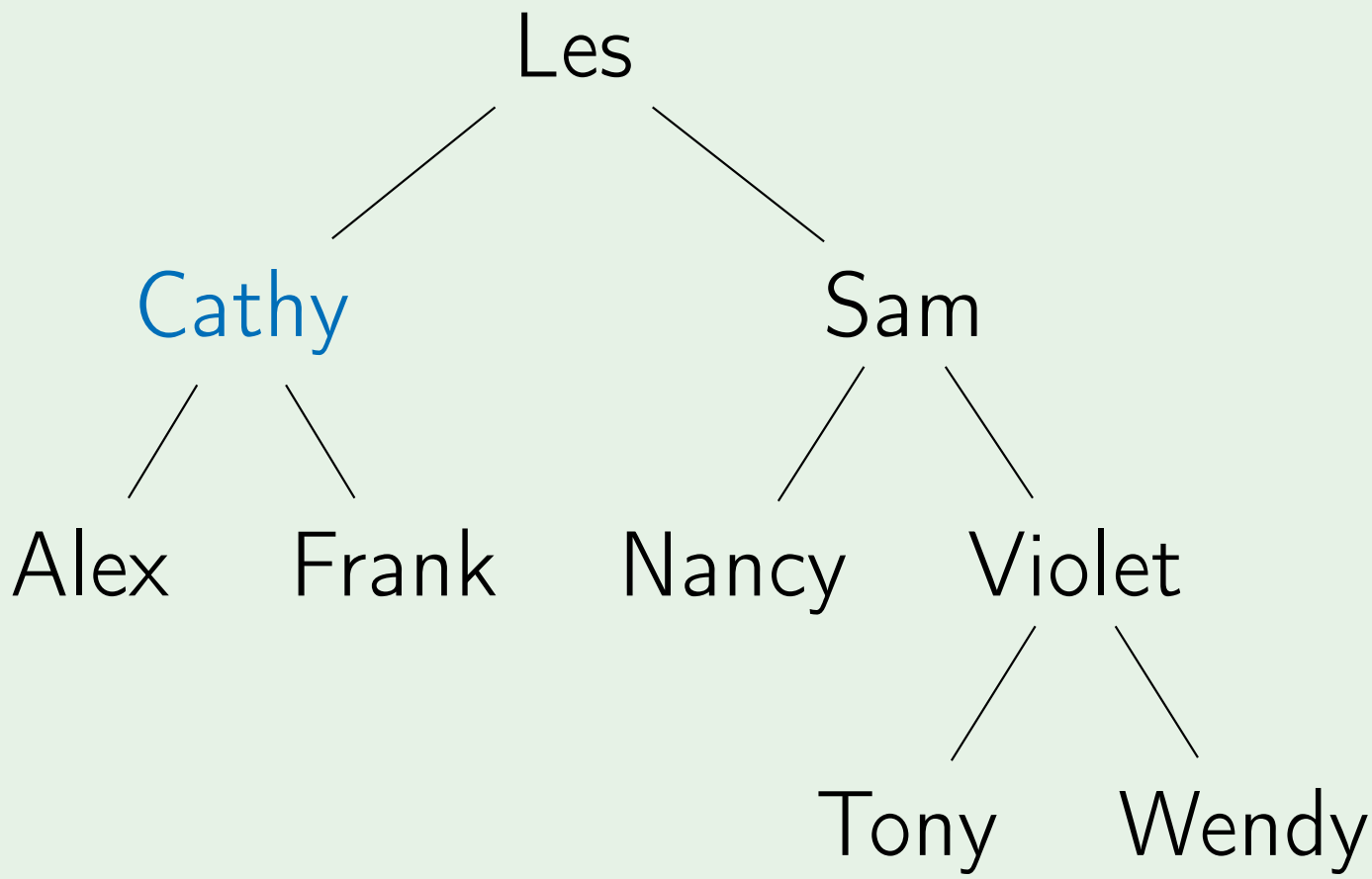
LevelTraversal



Output: Les

Queue: Sam

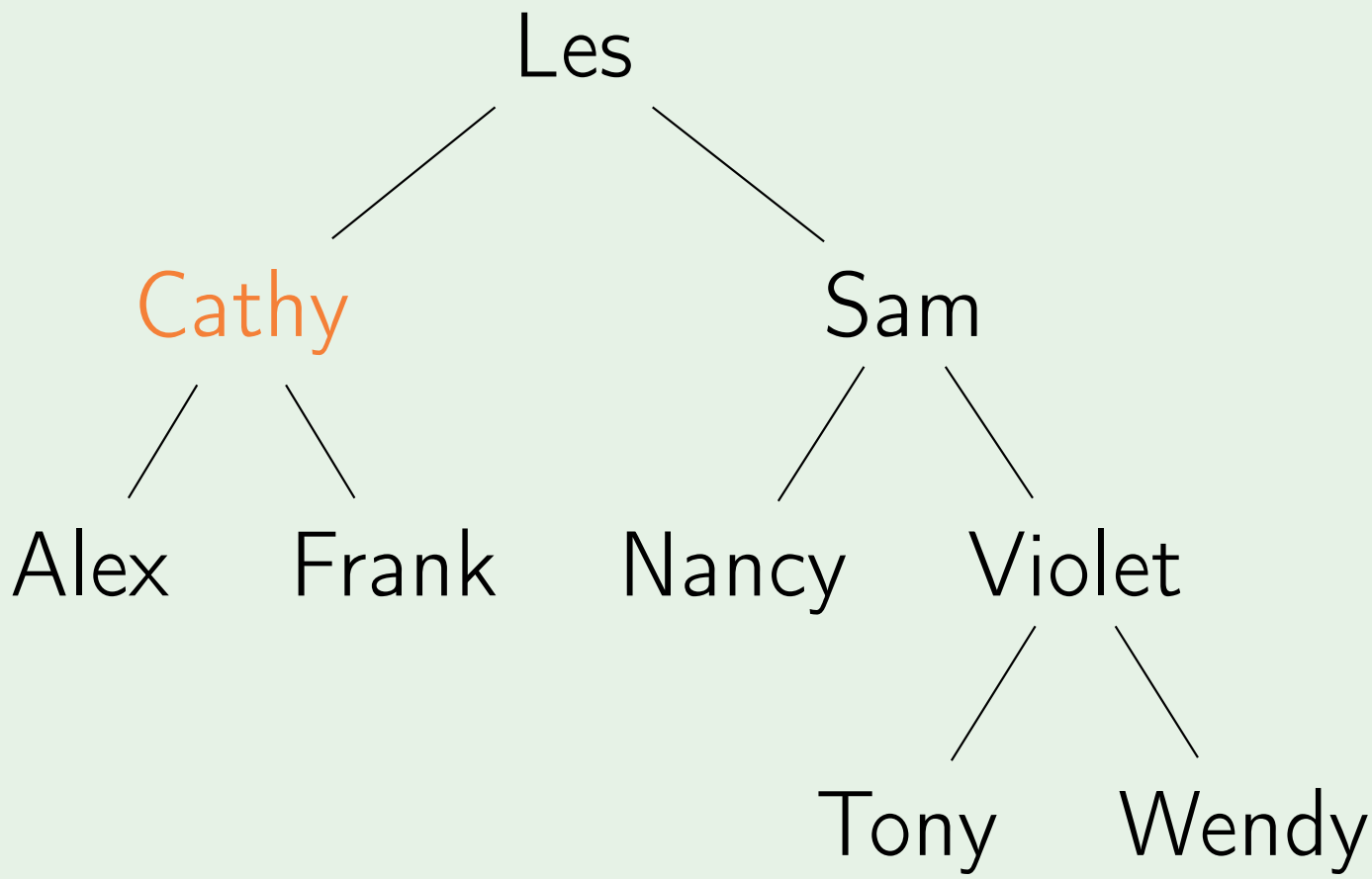
LevelTraversal



Output: Les Cathy

Queue: Sam

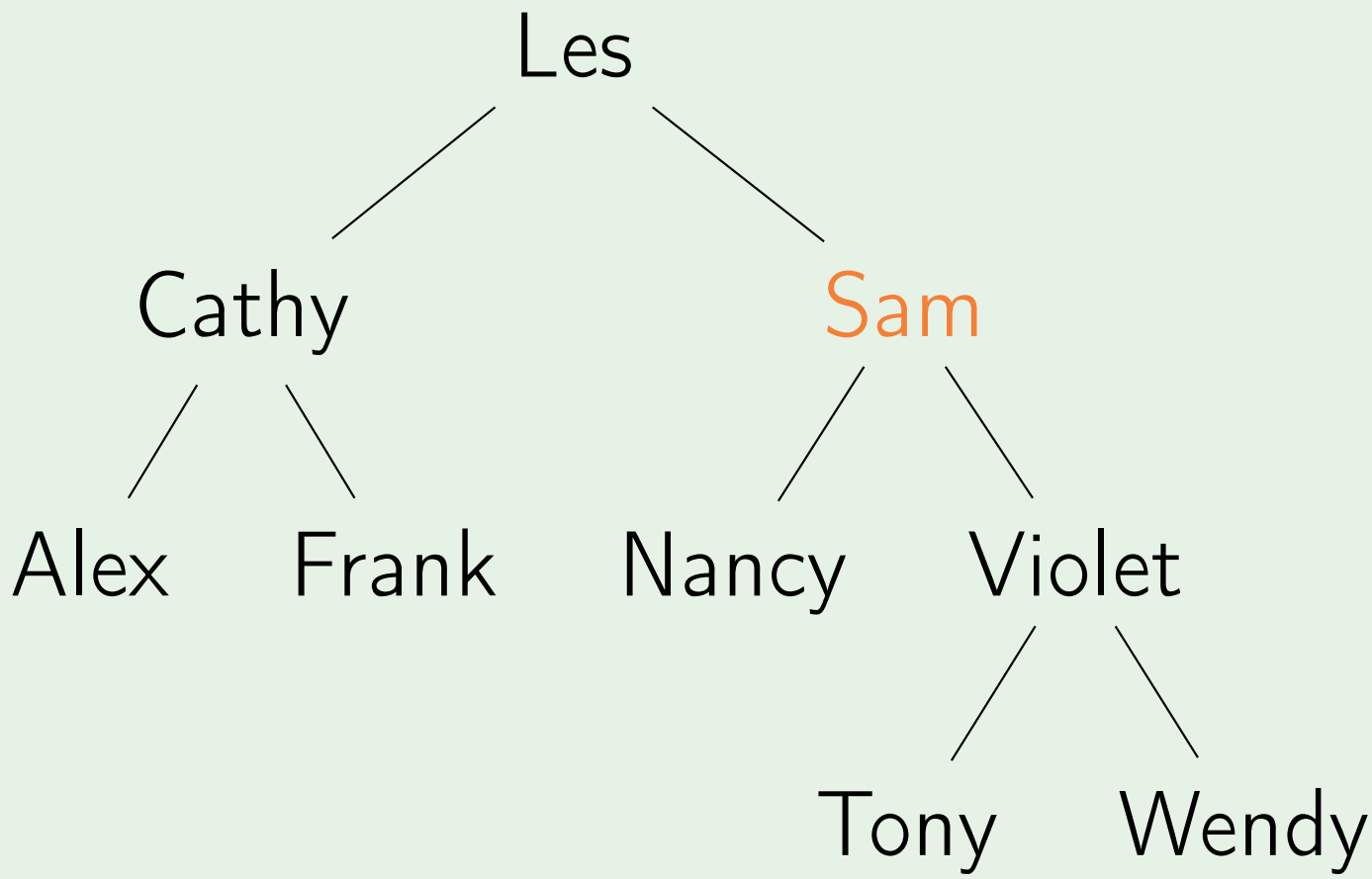
LevelTraversal



Output: Les Cathy

Queue: Sam,Alex, Frank

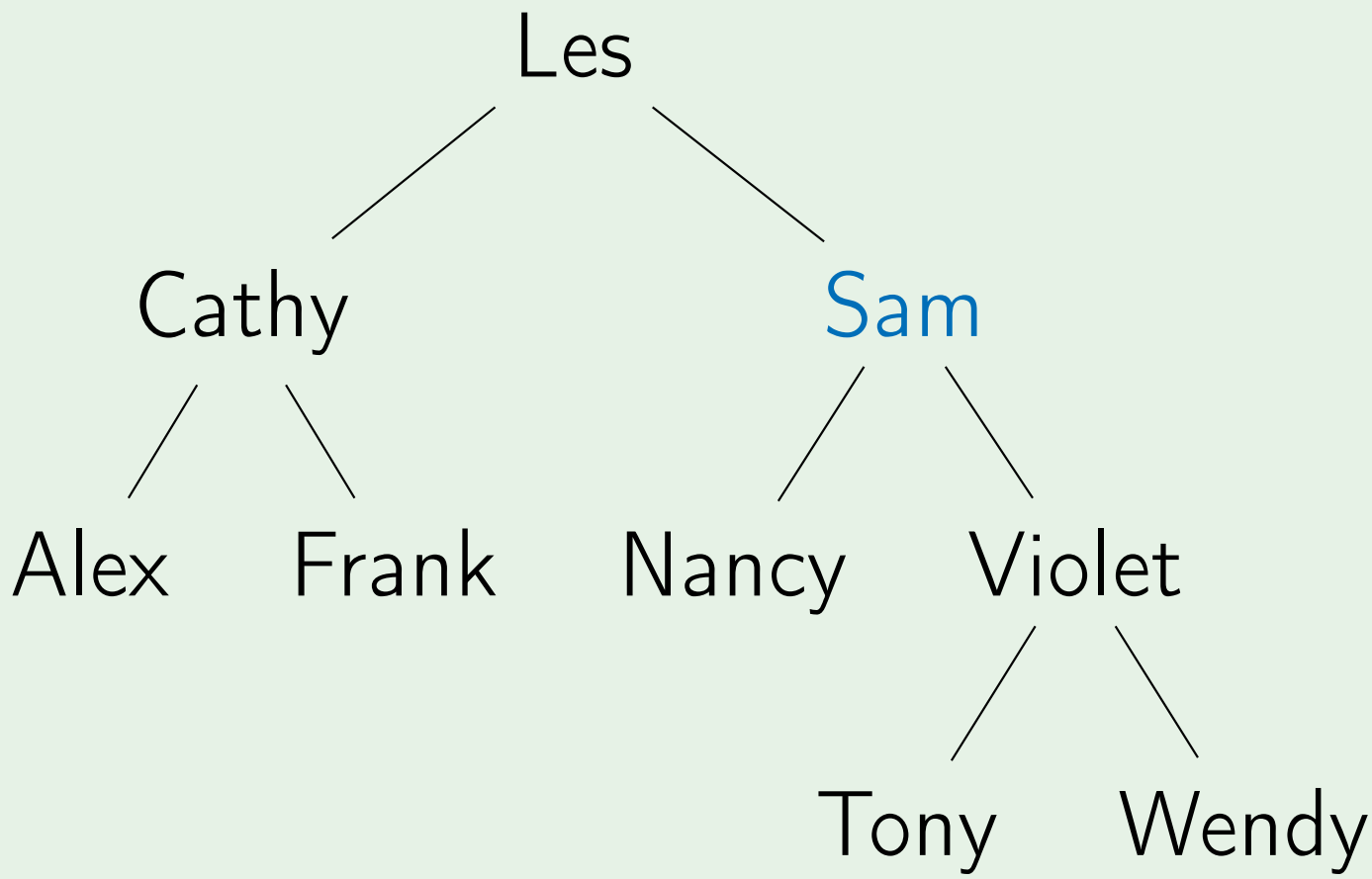
LevelTraversal



Output: Les Cathy

Queue: Alex, Frank

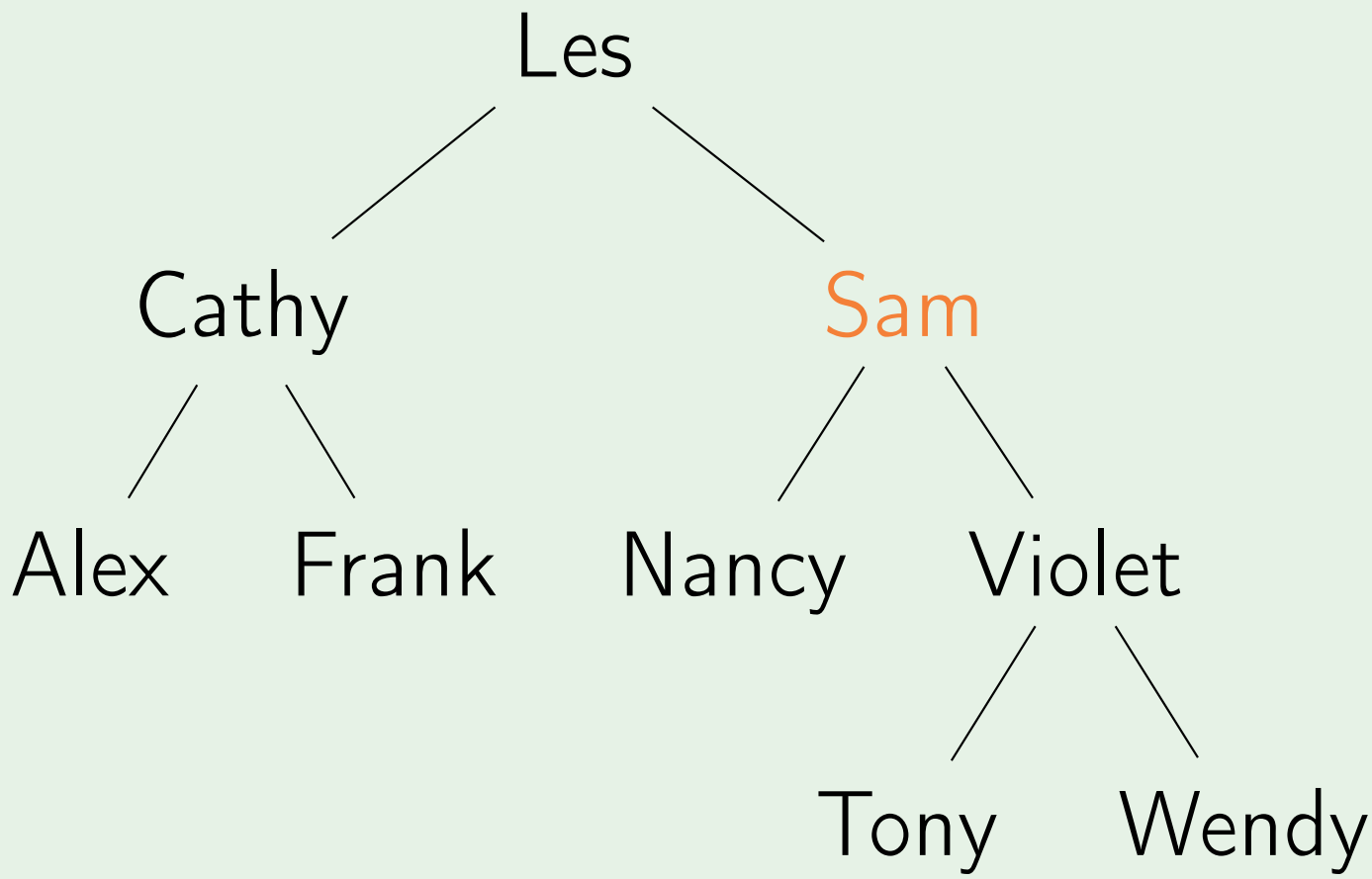
LevelTraversal



Output: Les Cathy Sam

Queue: Alex, Frank

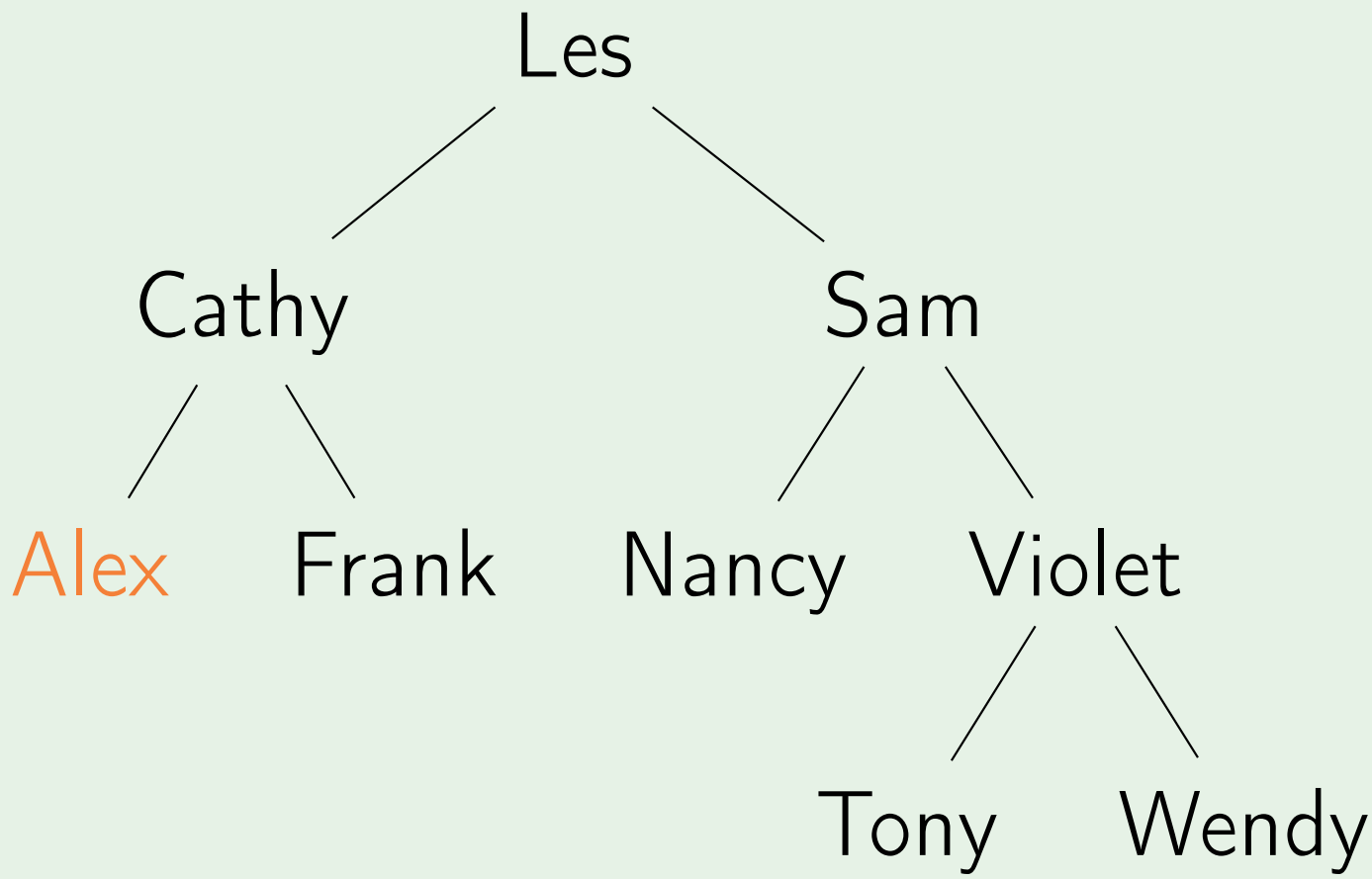
LevelTraversal



Output: Les Cathy Sam

Queue: Alex, Frank, Nancy, Violet

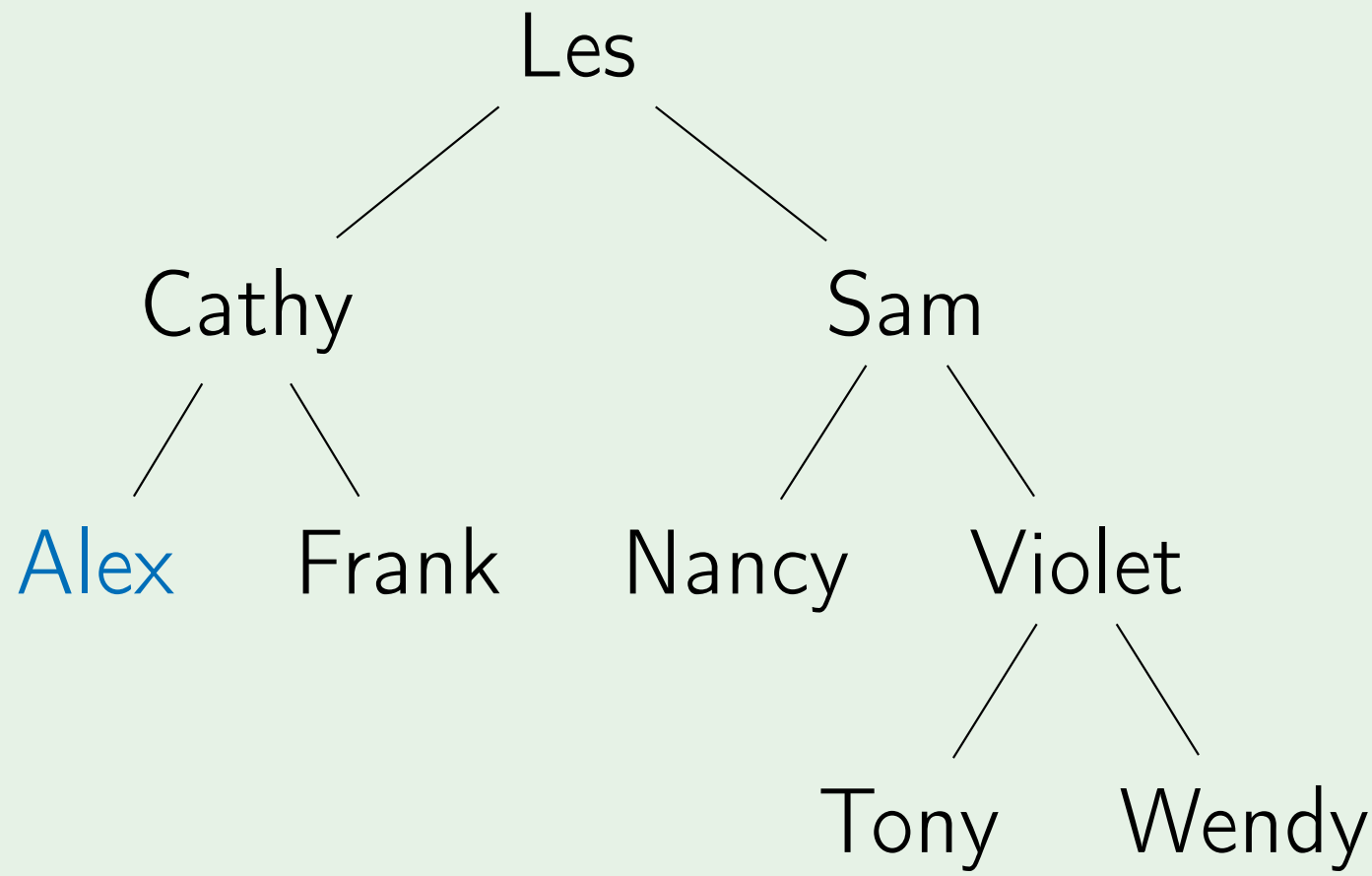
LevelTraversal



Output: Les Cathy Sam

Queue: Frank, Nancy, Violet

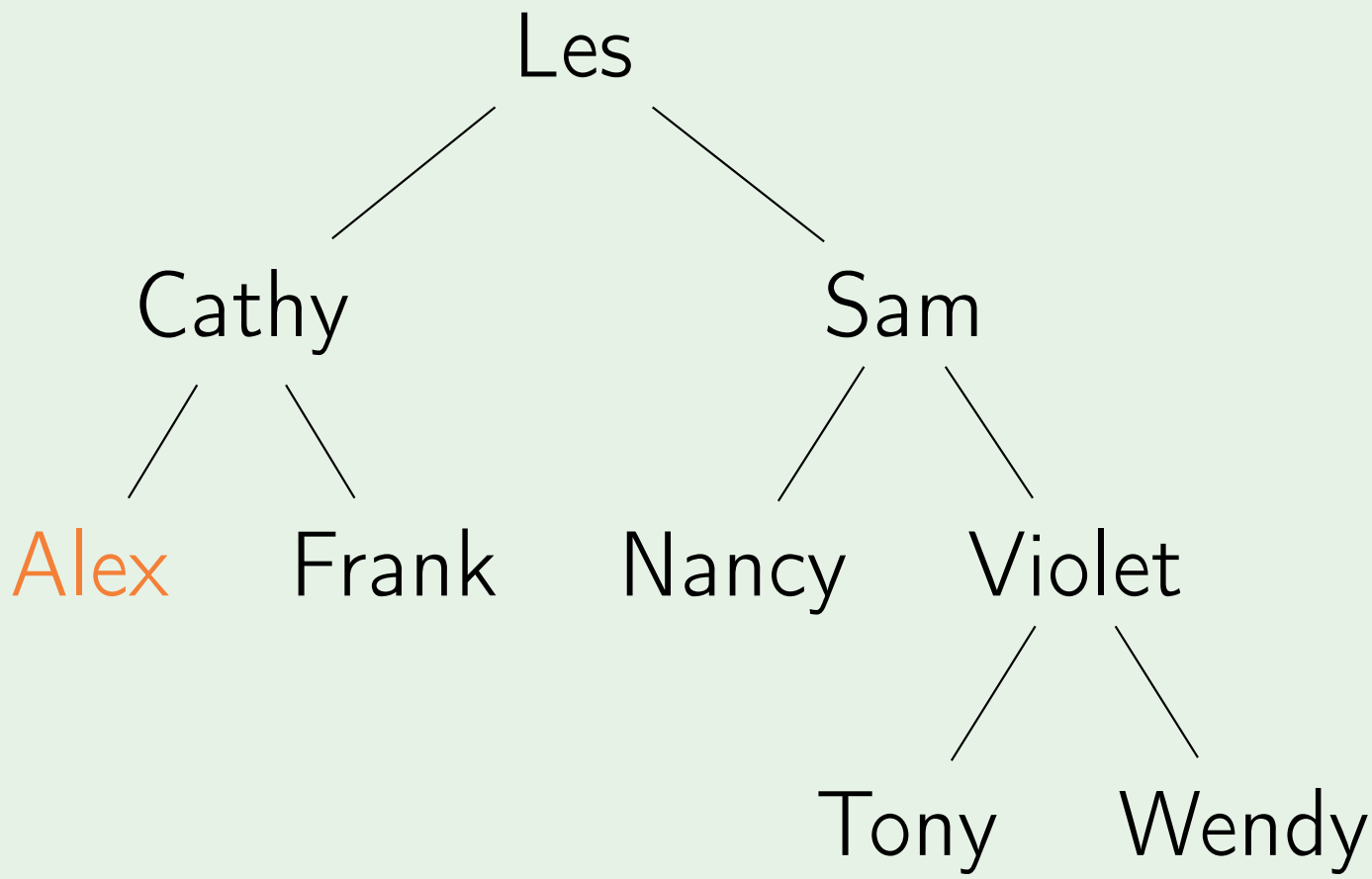
LevelTraversal



Output: Les Cathy Sam Alex

Queue: Frank, Nancy, Violet

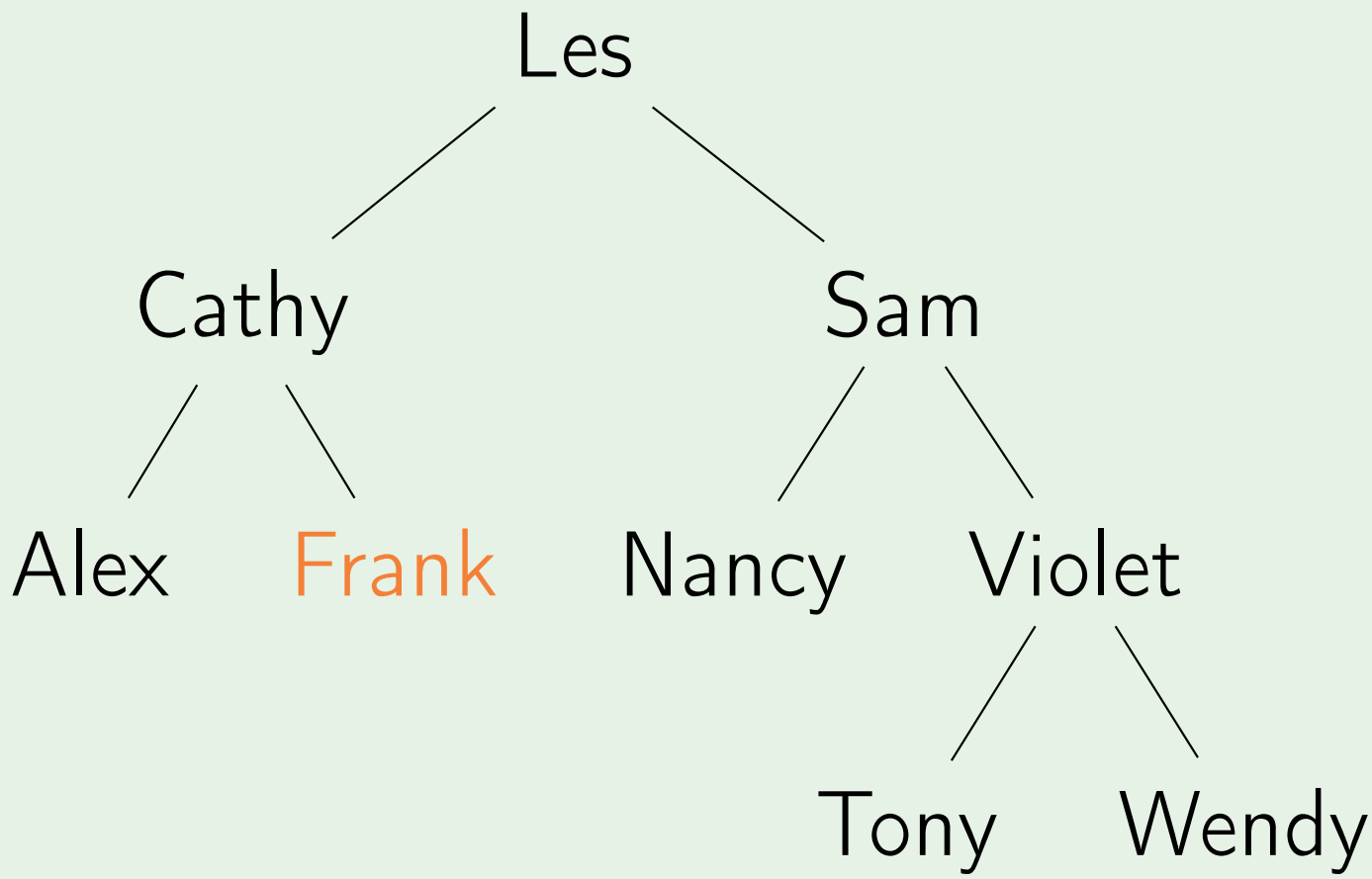
LevelTraversal



Output: Les Cathy Sam Alex

Queue: Frank, Nancy, Violet

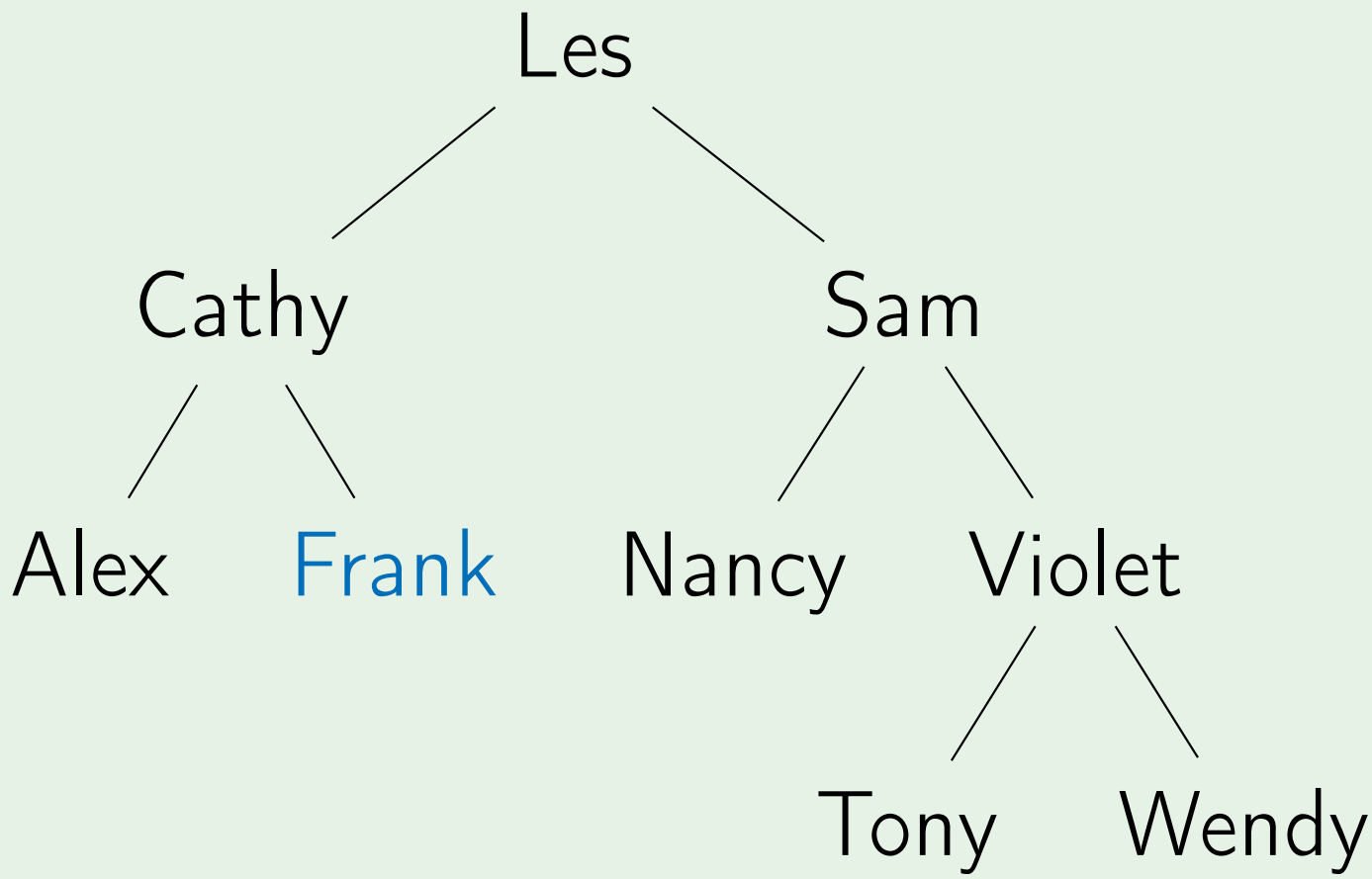
LevelTraversal



Output: Les Cathy Sam Alex

Queue: Nancy, Violet

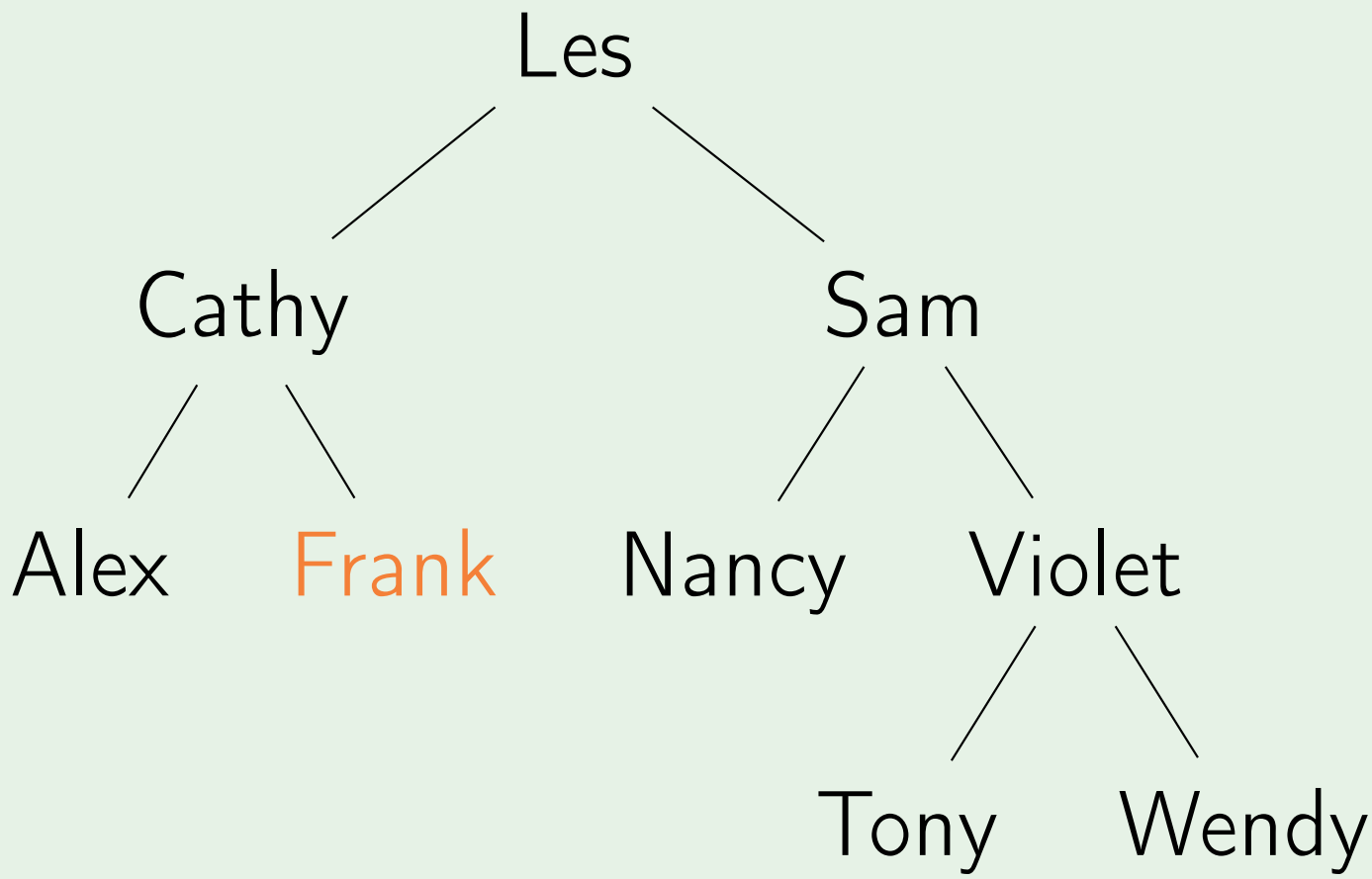
LevelTraversal



Output: Les Cathy Sam Alex Frank

Queue: Nancy, Violet

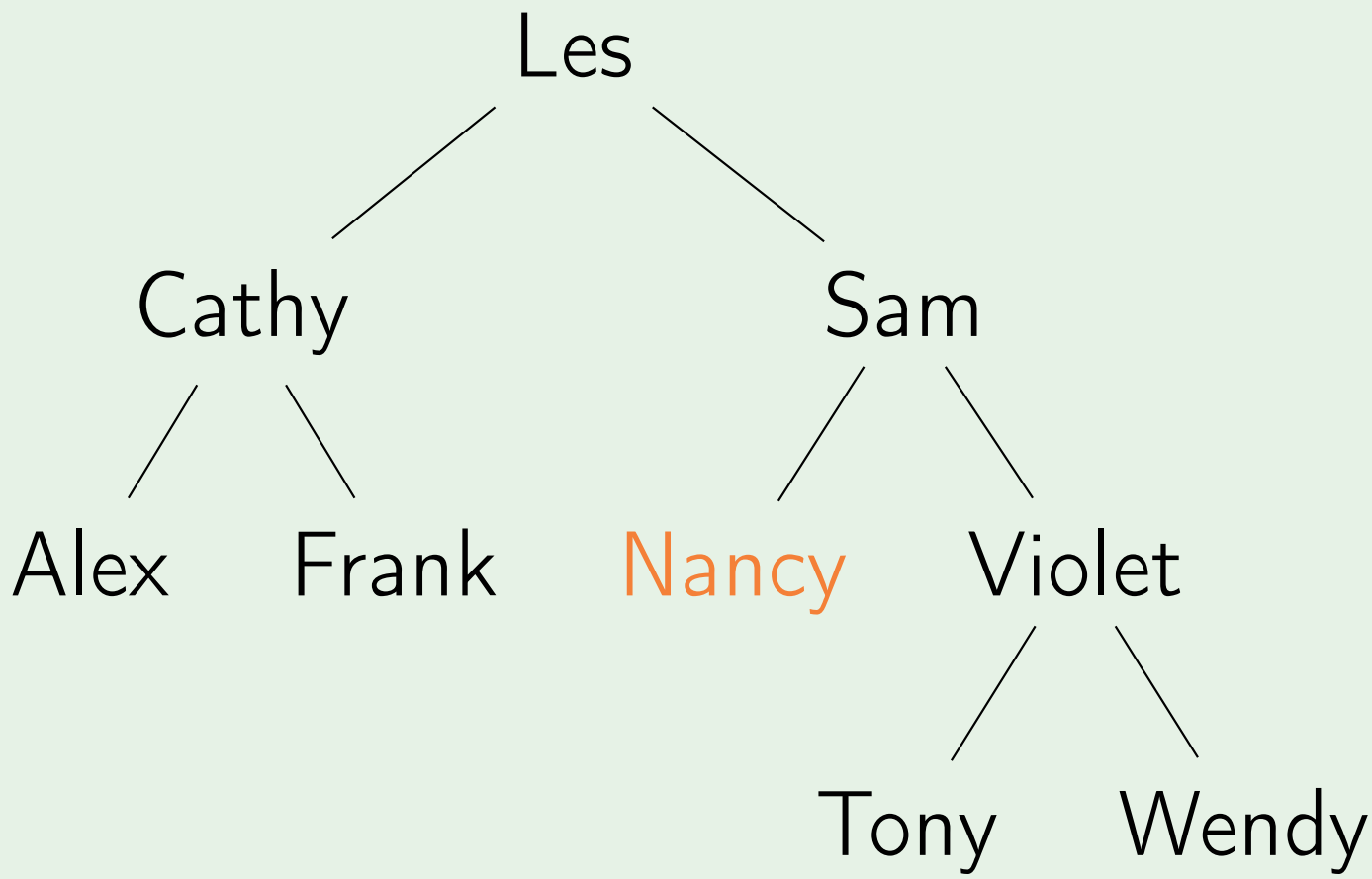
LevelTraversal



Output: Les Cathy Sam Alex Frank

Queue: Nancy, Violet

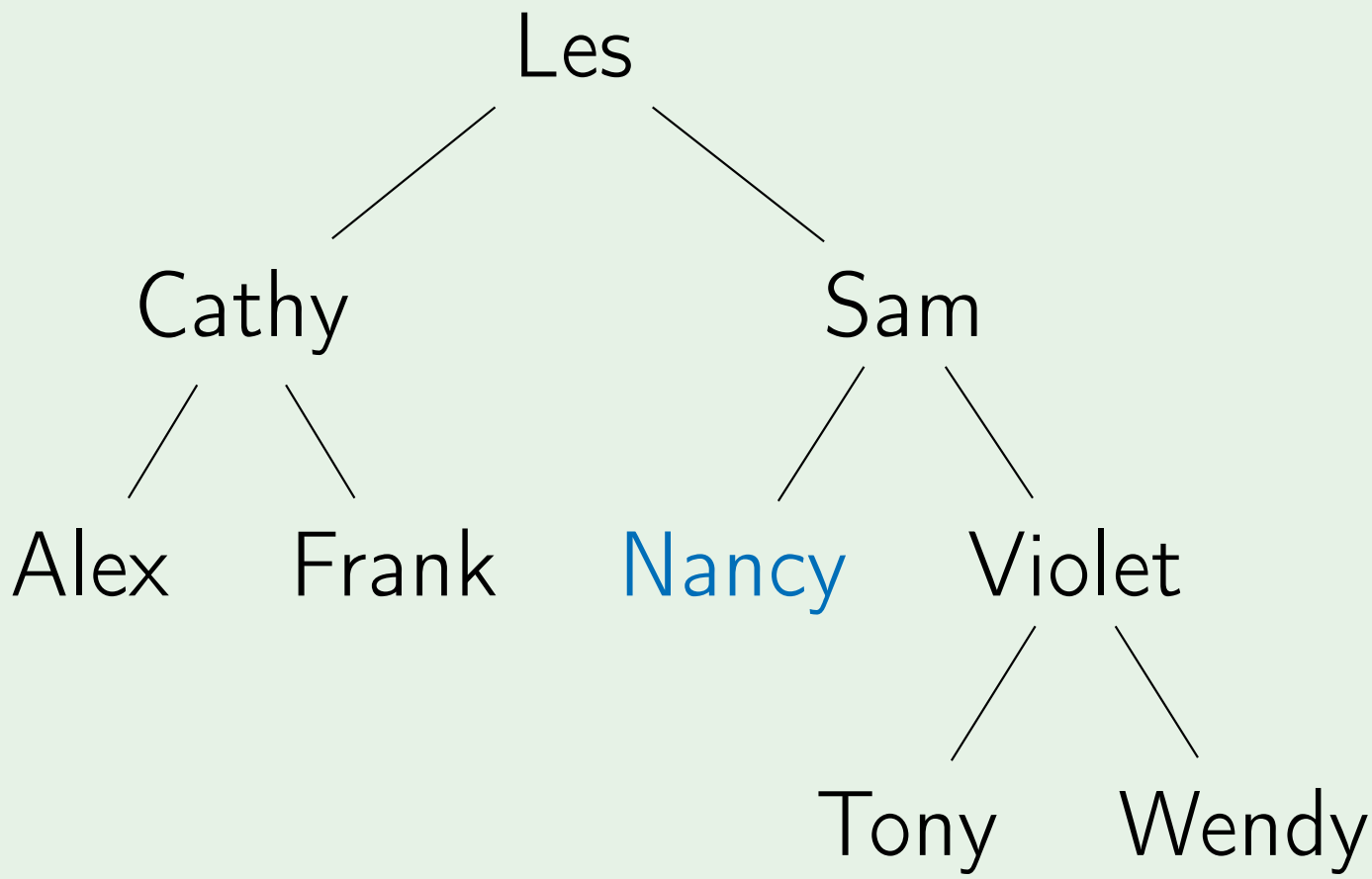
LevelTraversal



Output: Les Cathy Sam Alex Frank

Queue: Violet

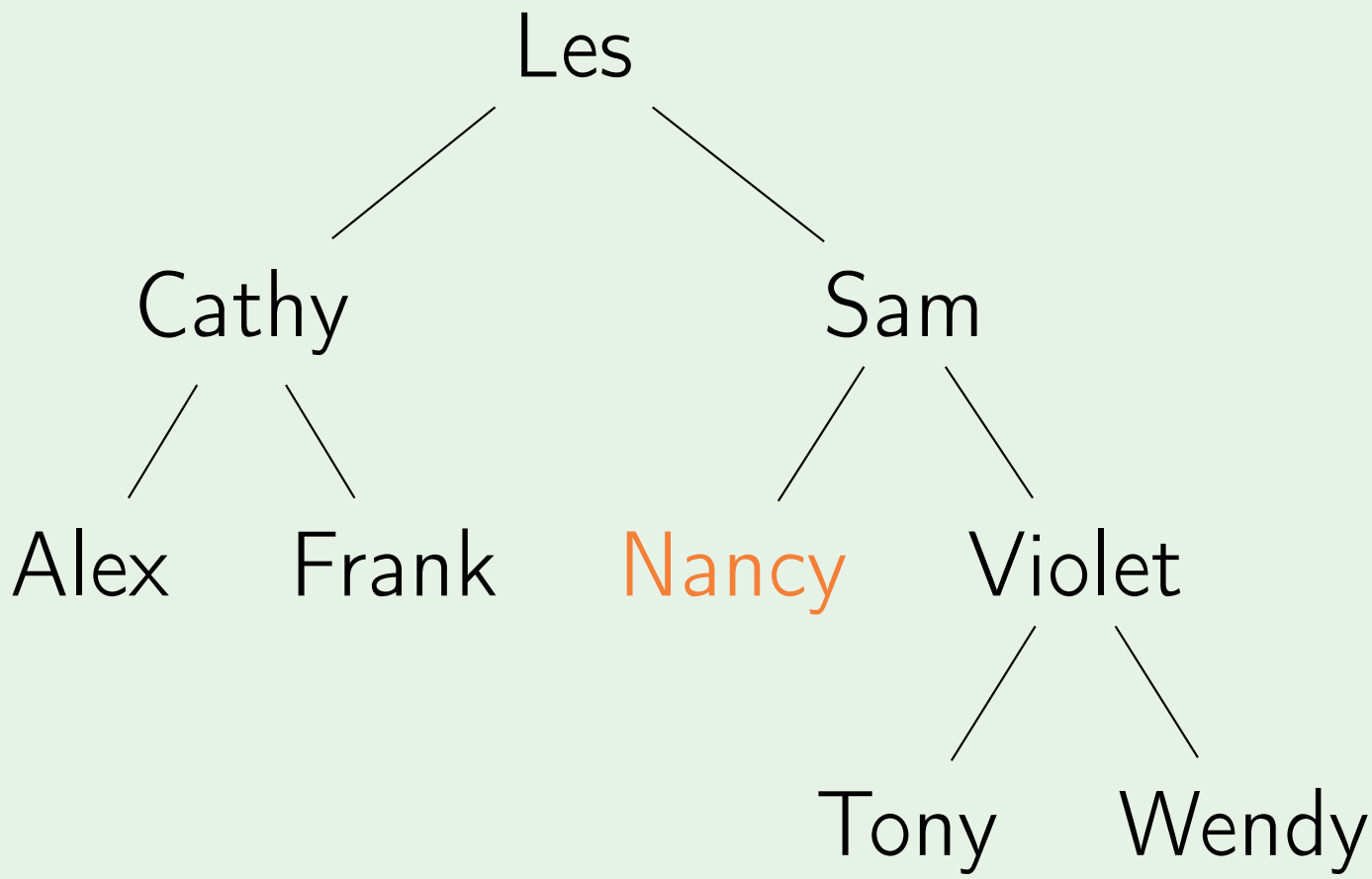
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy

Queue: Violet

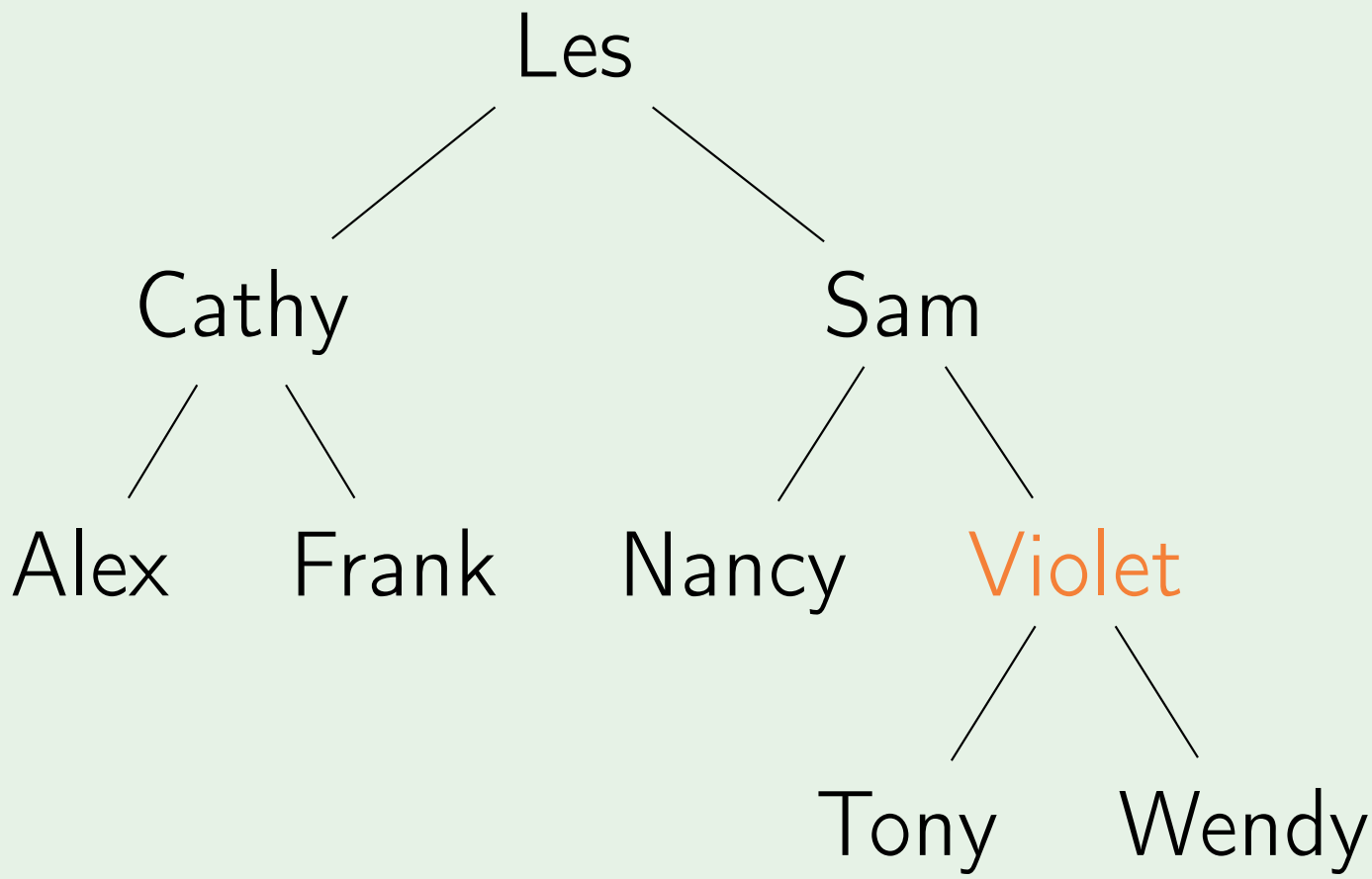
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy

Queue: Violet

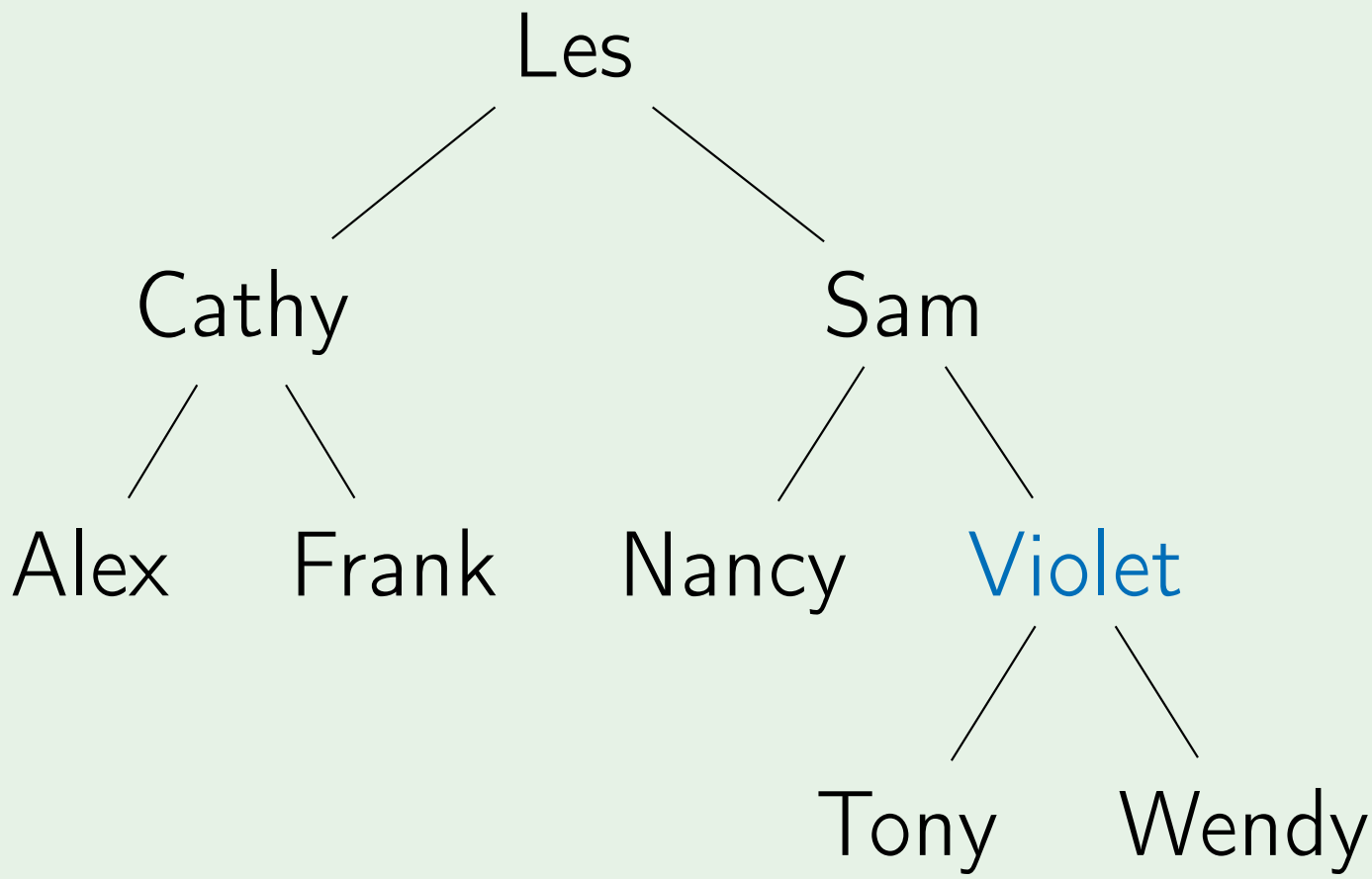
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy

Queue:

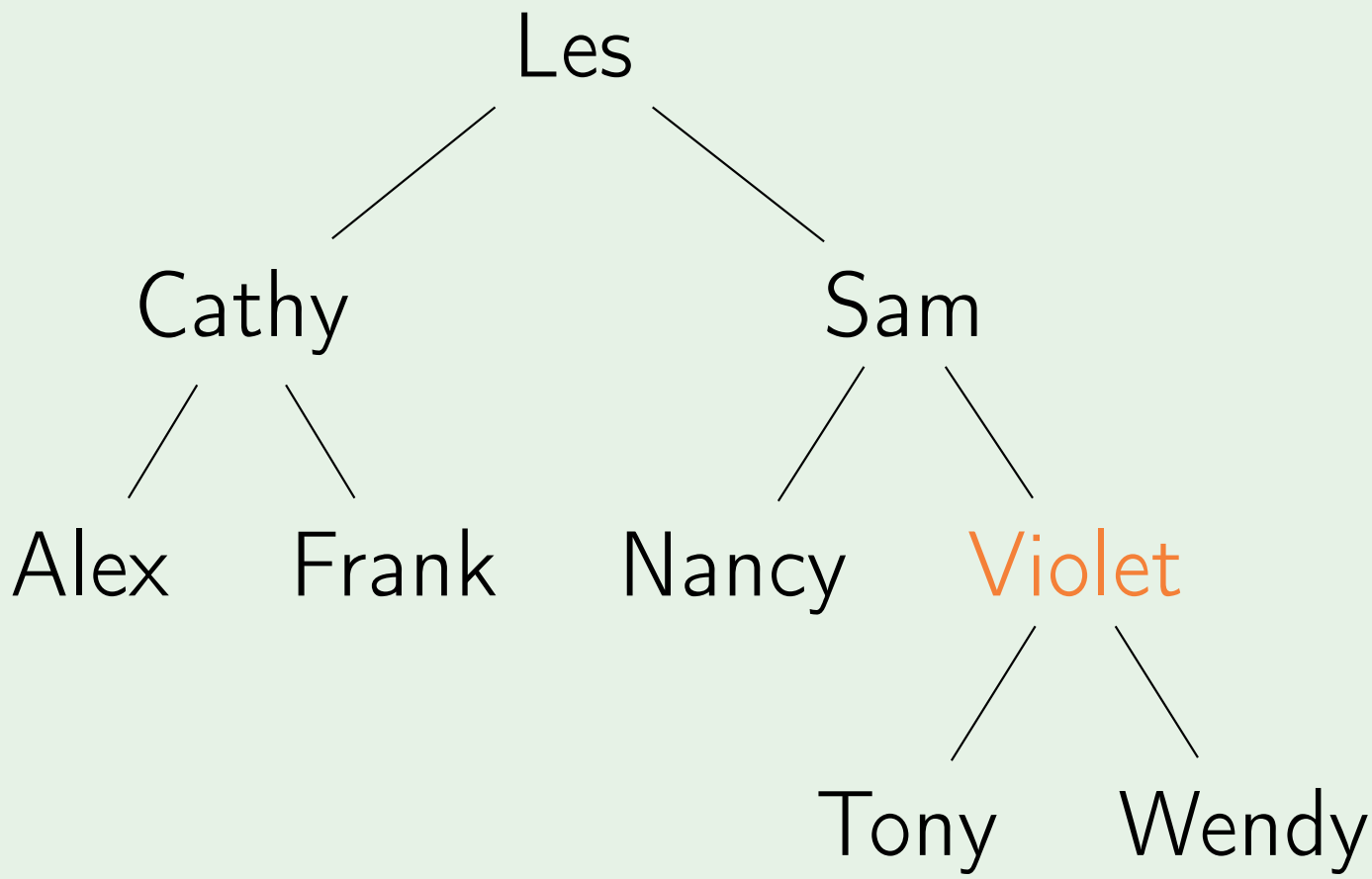
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet

Queue:

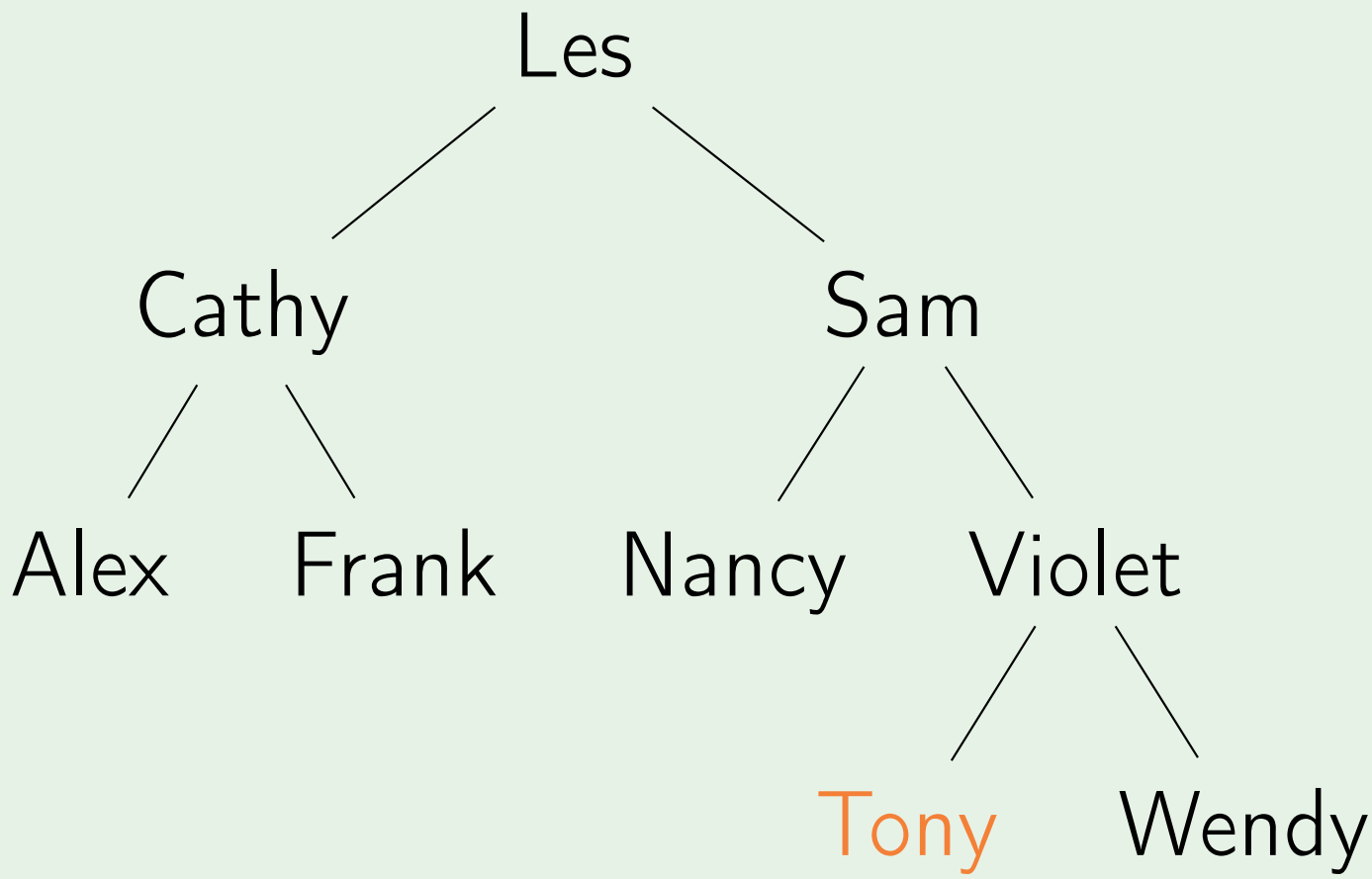
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet

Queue: Tony Wendy

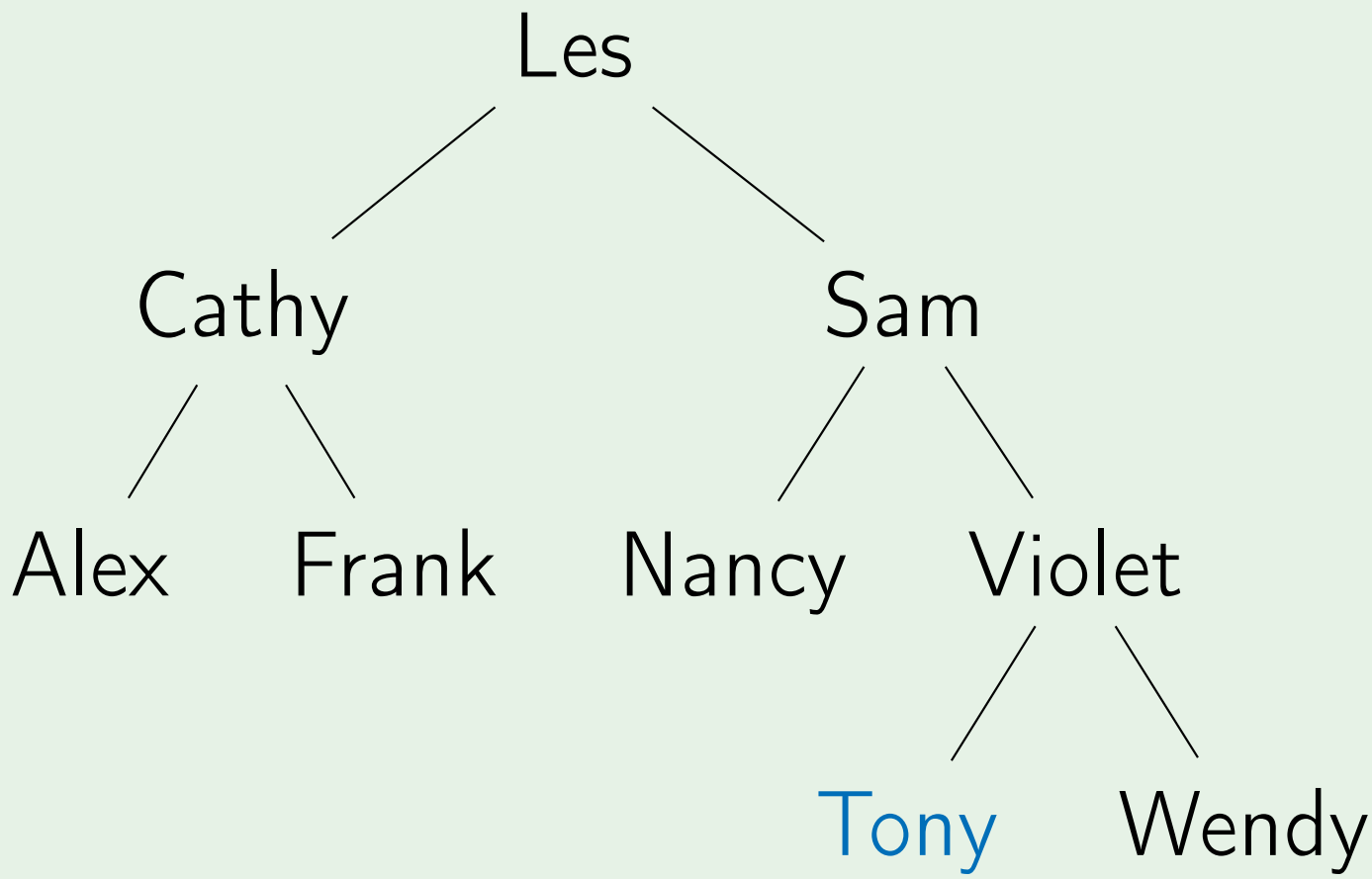
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet

Queue: Wendy

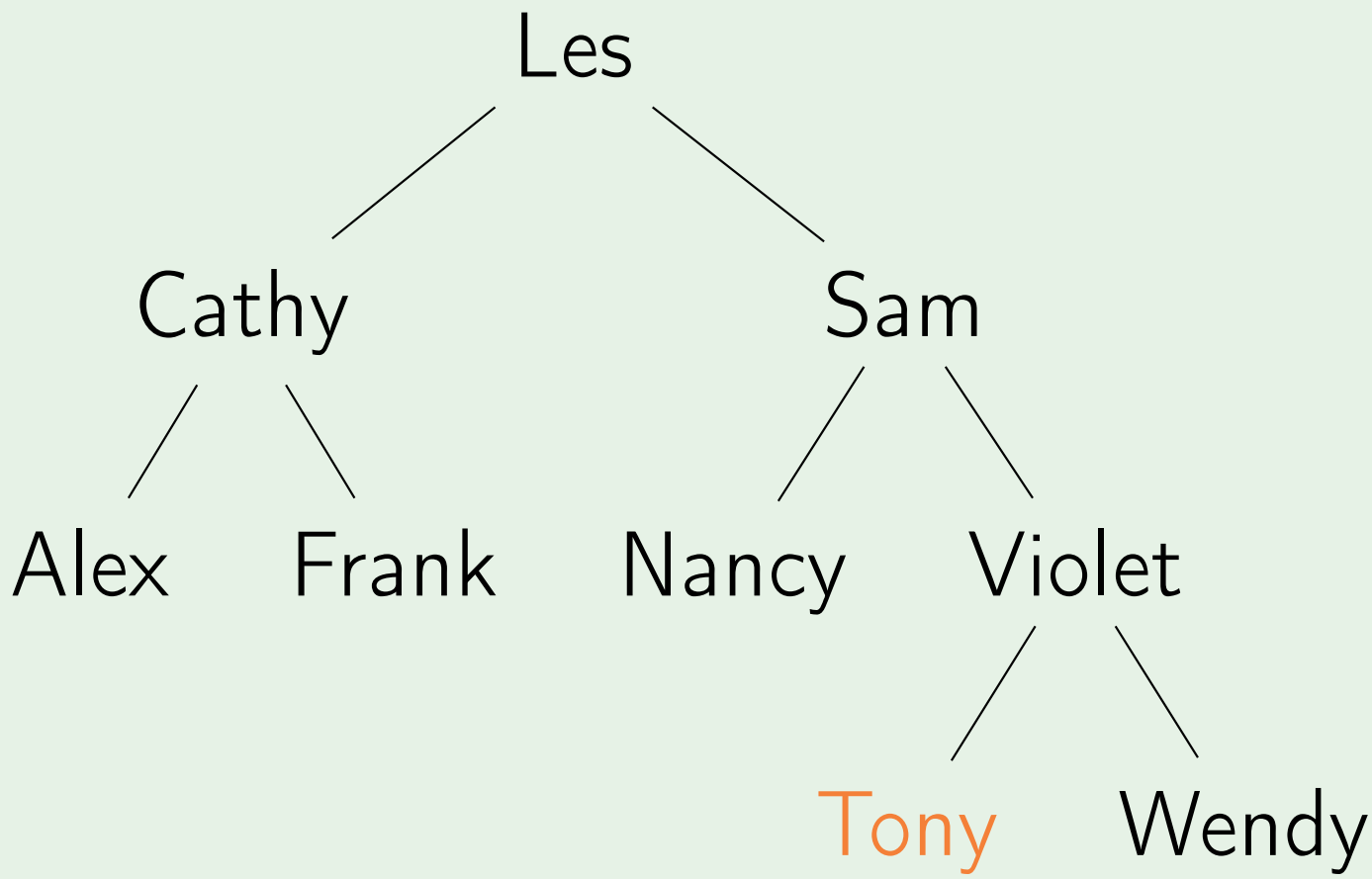
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet Tony

Queue: Wendy

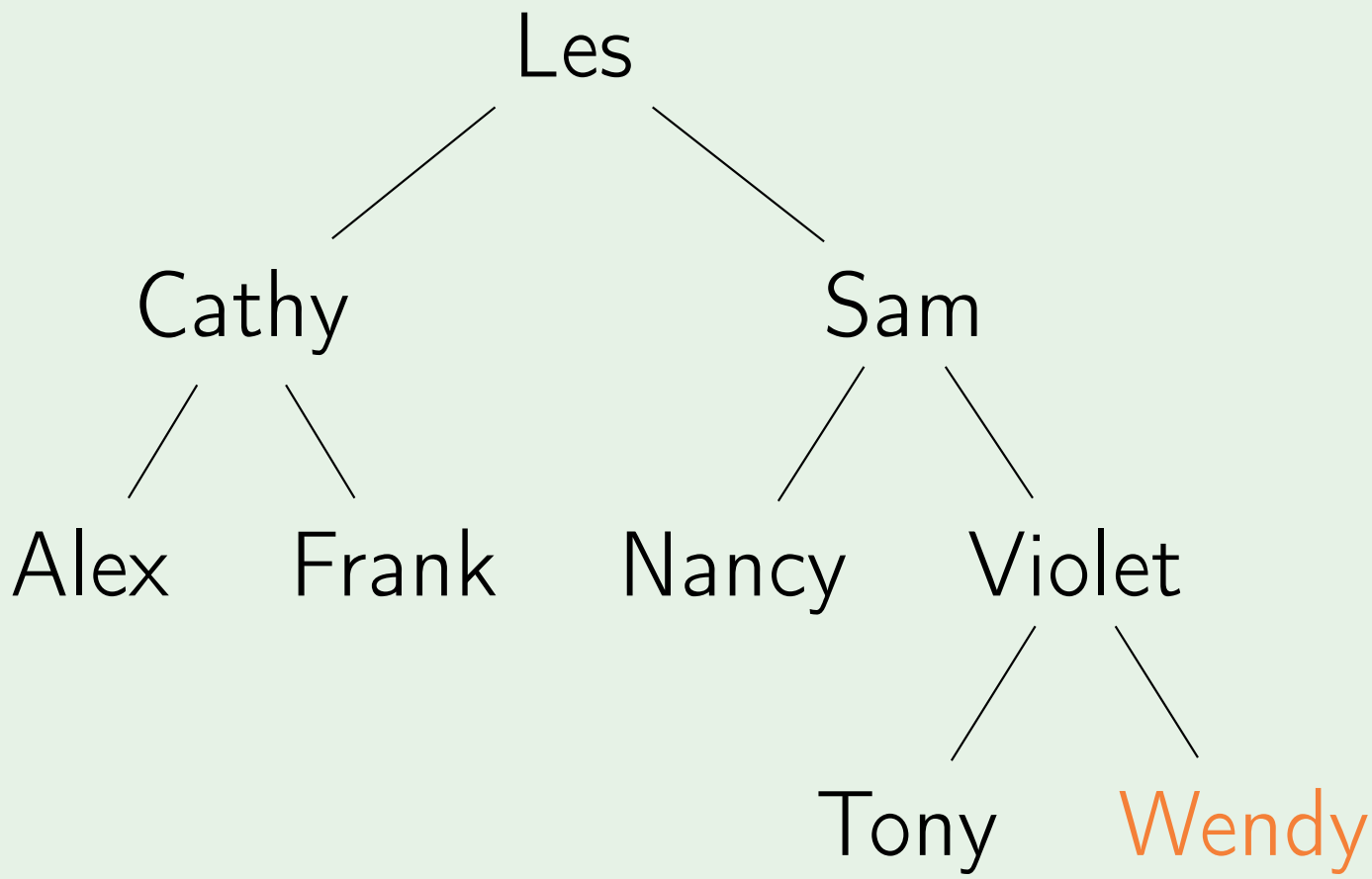
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet Tony

Queue: Wendy

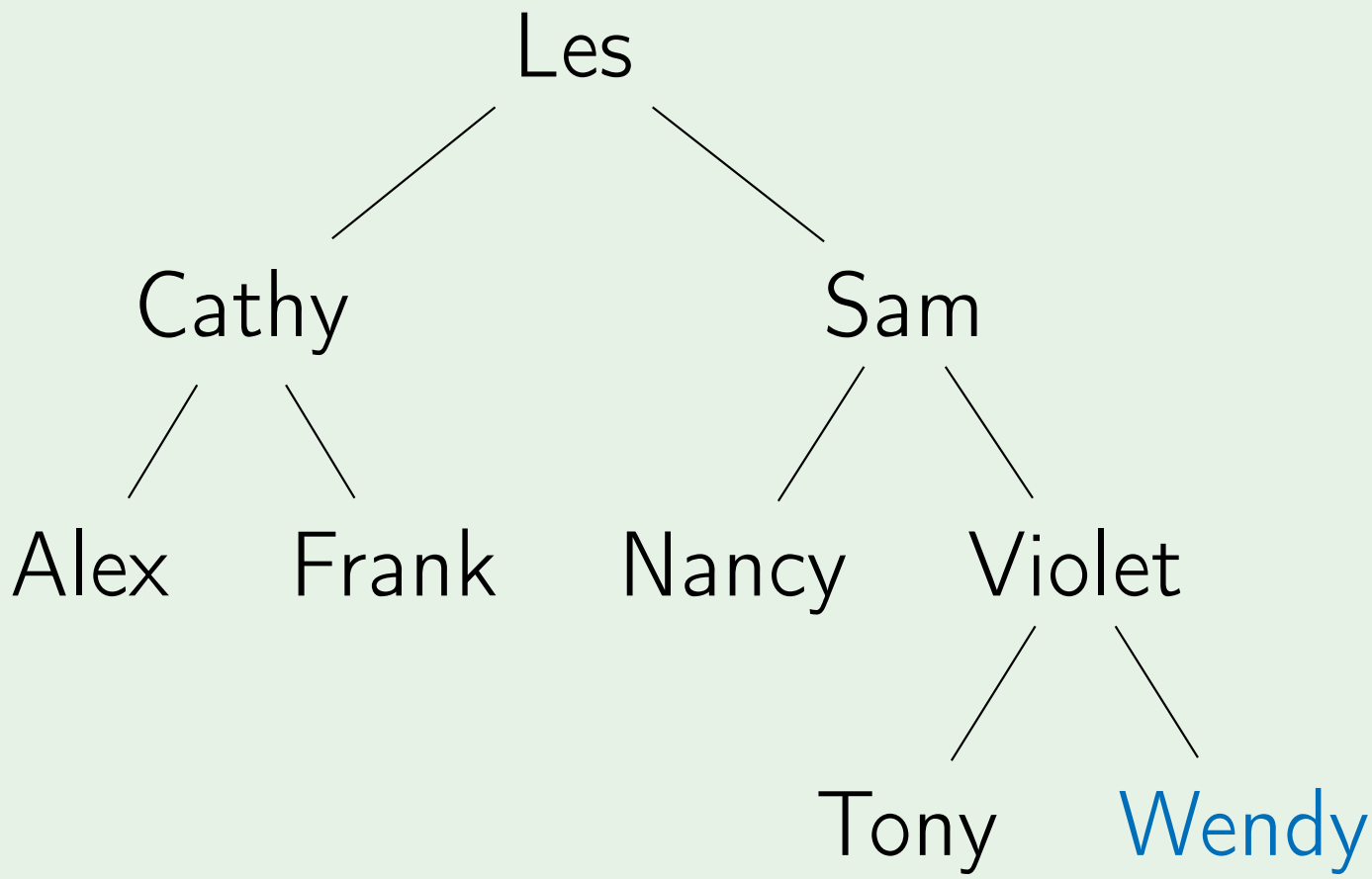
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet Tony

Queue:

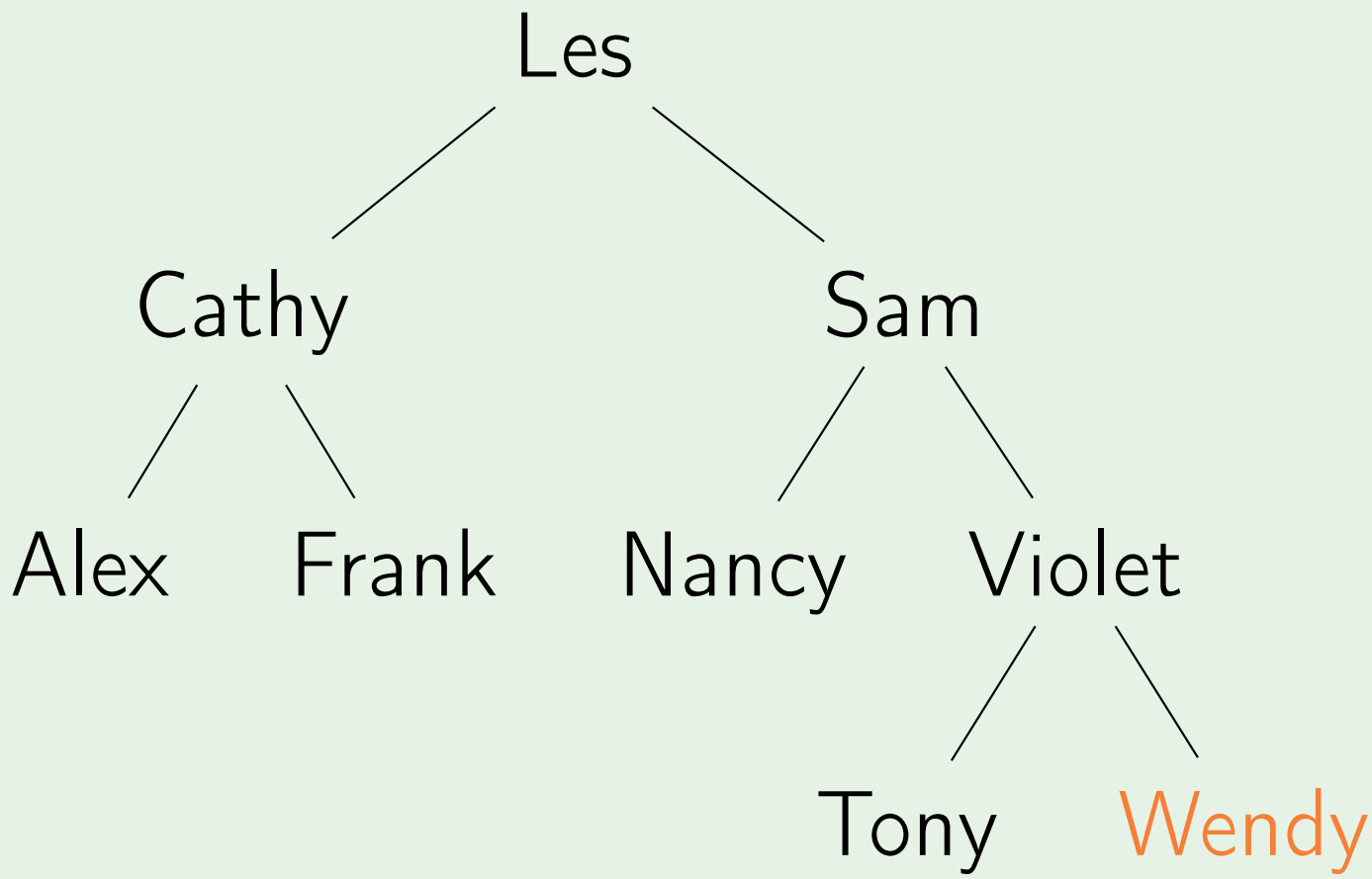
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet Tony Wendy

Queue:

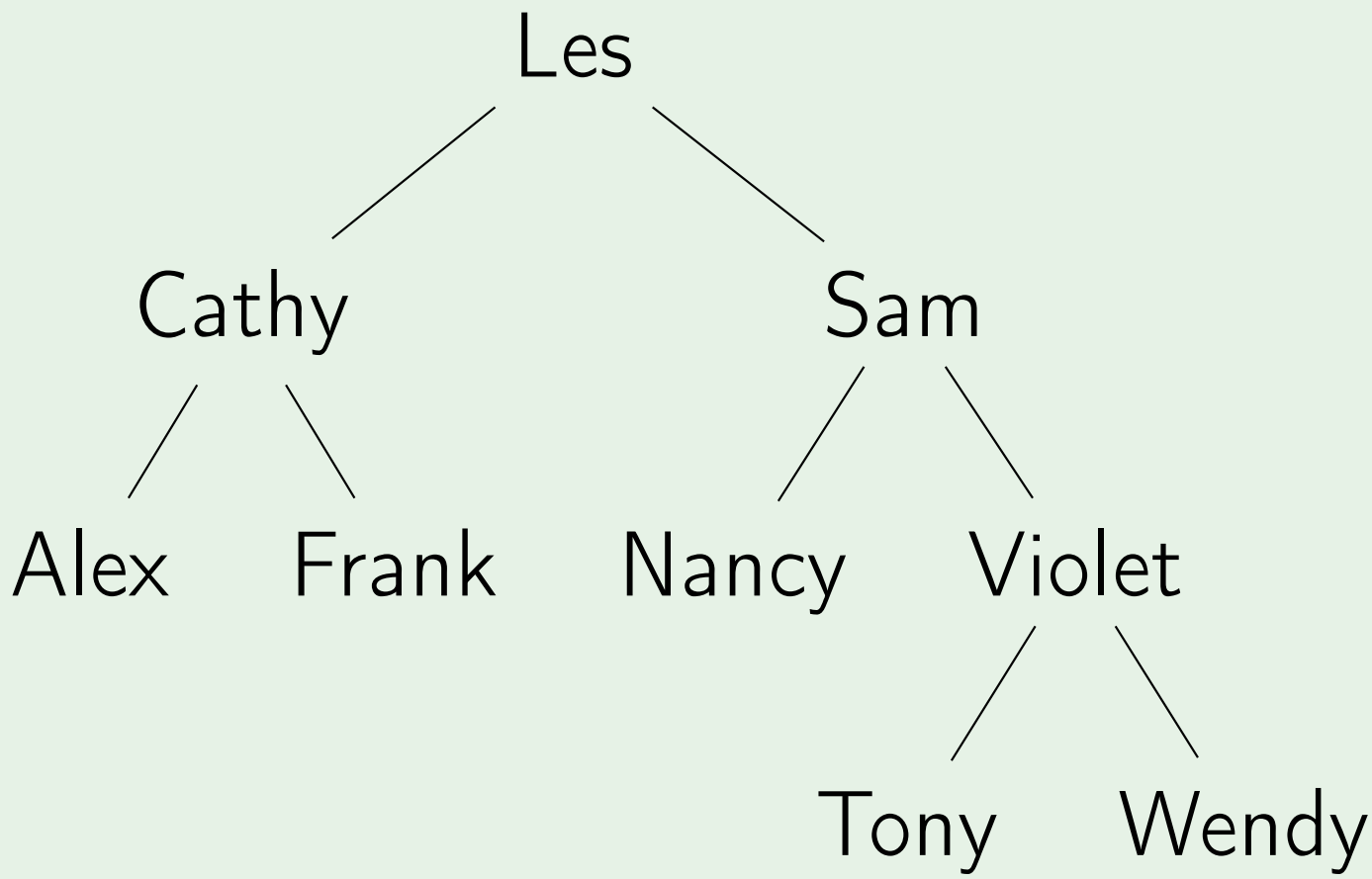
LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet Tony Wendy

Queue:

LevelTraversal



Output: Les Cathy Sam Alex Frank Nancy Violet Tony Wendy

Queue:

Summary

- Trees are used for lots of different things.

Summary

- Trees are used for lots of different things.
- Trees have a key and children.

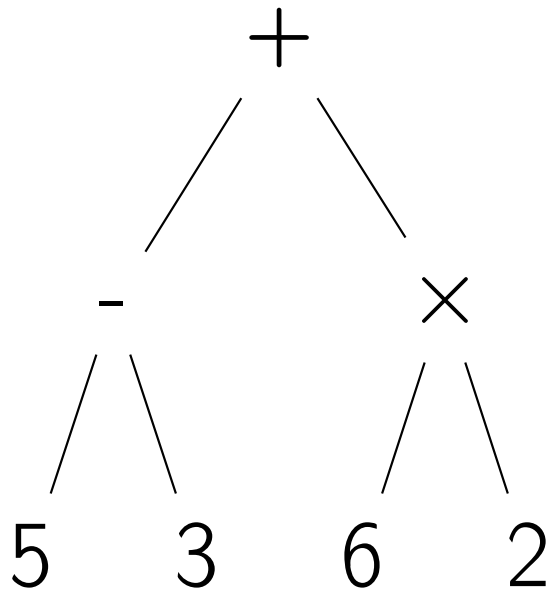
Summary

- Trees are used for lots of different things.
- Trees have a key and children.
- Tree walks: DFS (pre-order, in-order, post-order) and BFS.

Summary

- Trees are used for lots of different things.
- Trees have a key and children.
- Tree walks: DFS (pre-order, in-order, post-order) and BFS.
- When working with a tree, recursive algorithms are common.

Summary

- Trees are used for lots of different things.
- Trees have a key and children.
- Tree walks: DFS (pre-order, in-order, post-order) and BFS.
- When working with a tree, recursive algorithms are common.
- In Computer Science, trees grow down!

For Tree-traversal quiz