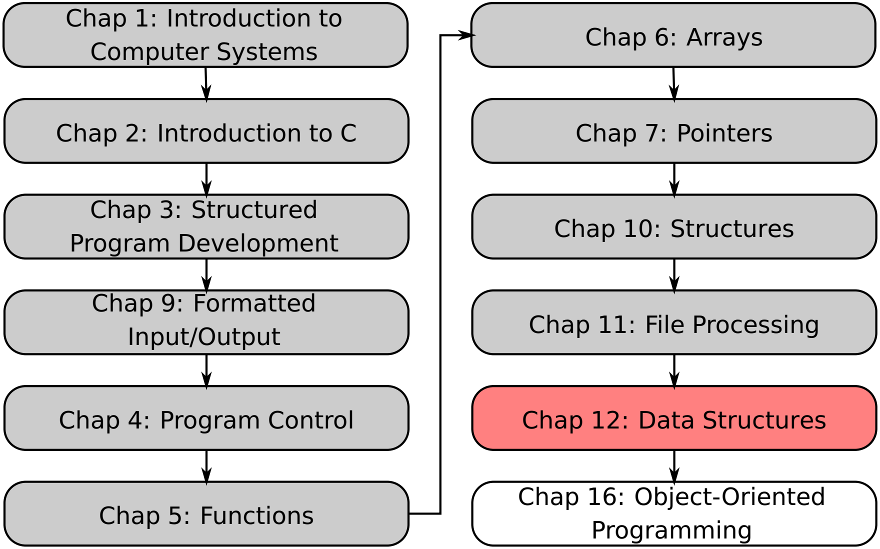
Computer Programming 143 – Lecture 30
Dynamic Data Structures III

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Module Overview


Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

Copyright & Disclaimer

Lecture Overview

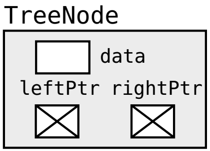
12.7 Trees I

Trees

- Nonlinear dynamic data structure with self-referential elements
- Elements contain 2 or more pointers to other elements
  - If each element contain only 2 pointers to other elements, it is called a *binary* tree

12.7 Trees II

```
struct treeNode {
  struct treeNode *leftPtr;
  int data;
  struct treeNode *rightPtr;
};
typedef struct treeNode TreeNode;
typedef TreeNode *TreeNodePtr;
```



Use of trees

- Efficient search and sorting algorithms
- Store and evaluation of mathematical expressions

12.7 Trees III

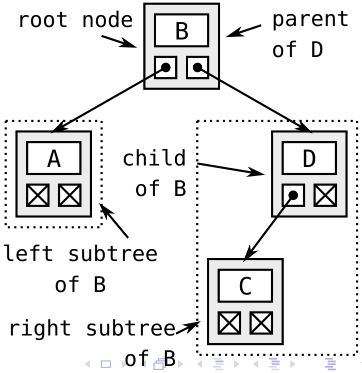
Concepts used in trees

Creating and handling trees require a combination of

- Structures
- Pointers
- Recursive functions

Terminology

- Root node: first node in a tree
- Left/right child: node to which the left/right pointer points
- Left/right subtree: left/right child and all its children
- Parent node: node of which specified node is one of its children

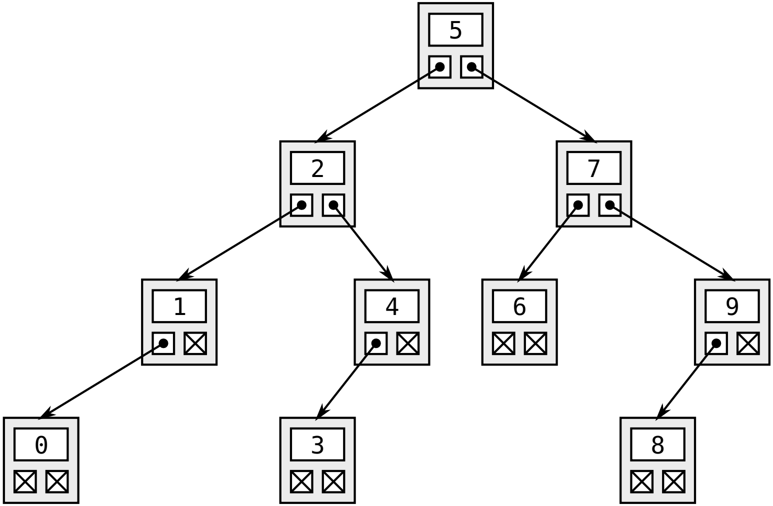


12.7 Sorted Binary Trees I

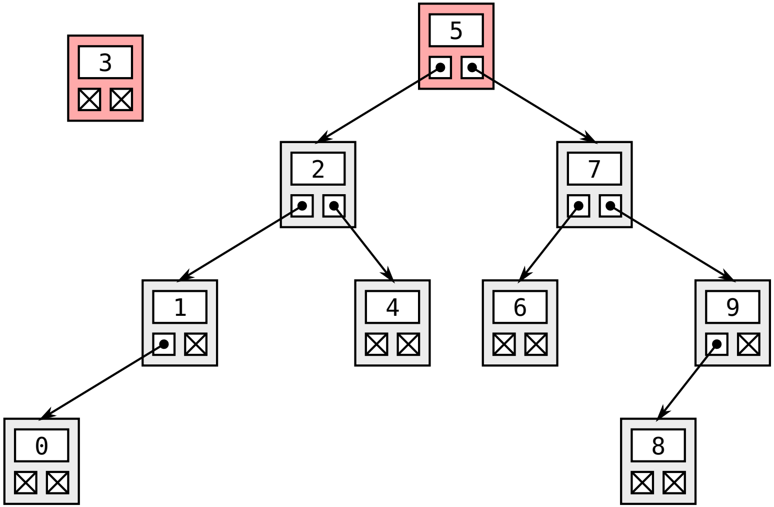
Sorted binary trees

- Binary tree – each node has 2 children
- Sorted:
  - All values in any left subtree are smaller than the value of its parent node
  - All values in any right subtree are greater than (greater or equal if we allow duplicates) the value of its parent node

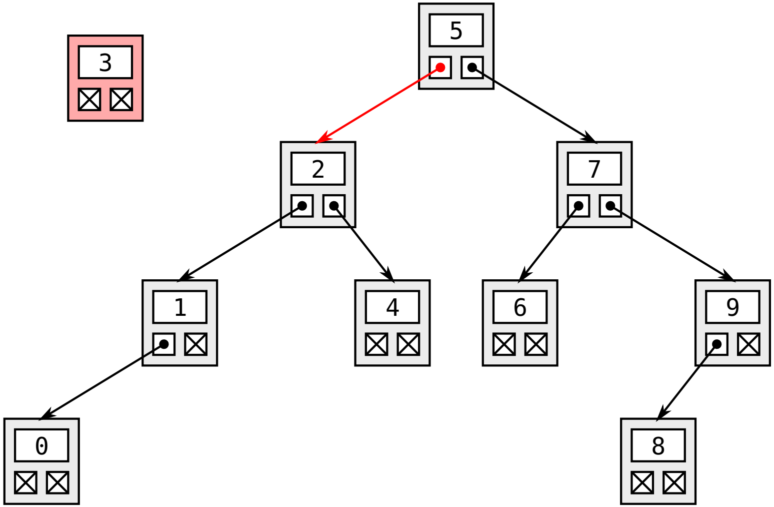
12.7 Sorted Binary Trees II



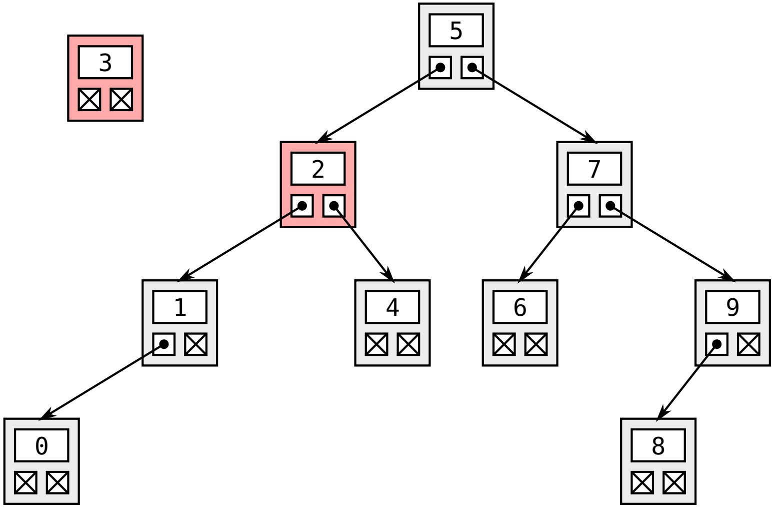
12.7 Insertion in Sorted Binary Trees



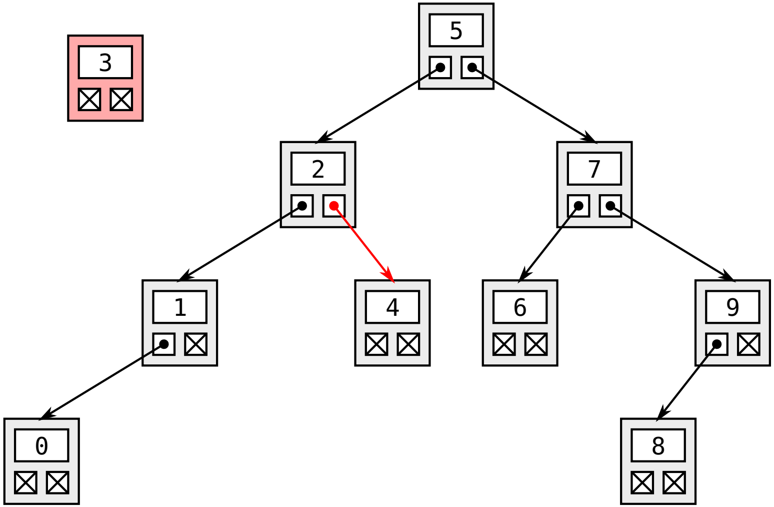
12.7 Insertion in Sorted Binary Trees



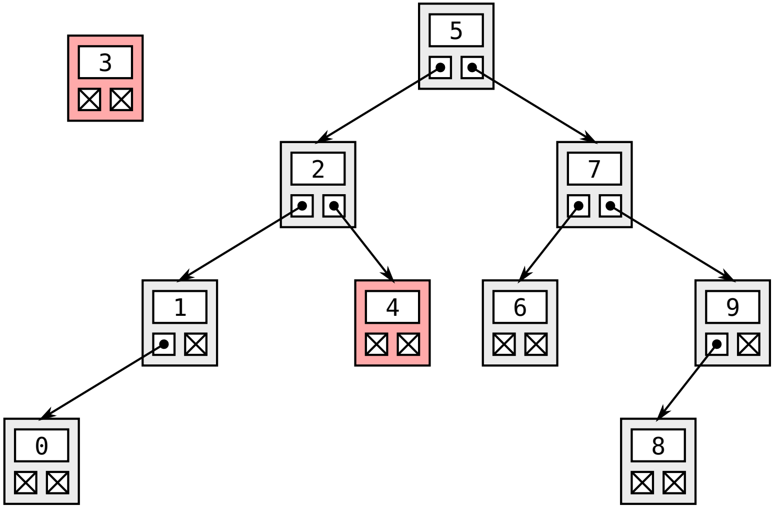
12.7 Insertion in Sorted Binary Trees



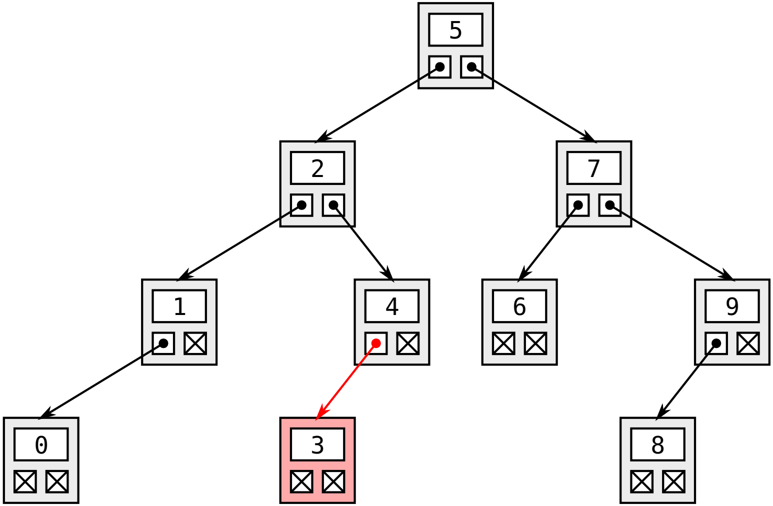
12.7 Insertion in Sorted Binary Trees



12.7 Insertion in Sorted Binary Trees



12.7 Insertion in Sorted Binary Trees



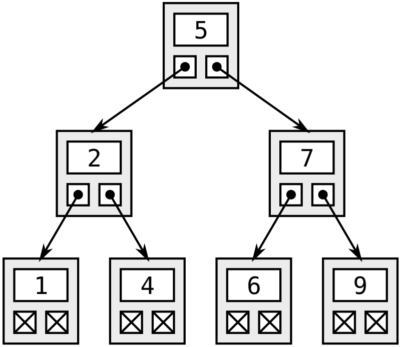
12.7 Traversal of Sorted Binary Trees I

Order of traversal of trees

- For some applications, all the nodes of the tree need to be visited
- The order in which the nodes are visited vary and may include:
  - Inorder: at each node,
    1. Traverse the node’s left subtree
    2. Process the node value
    3. Traverse the node’s right subtree
  - Preorder: at each node,
    1. Process the node value
    2. Traverse the node’s left subtree
    3. Traverse the node’s right subtree
  - Postorder: at each node,
    1. Traverse the node’s left subtree
    2. Traverse the node’s right subtree
    3. Process the node value
- Easily implemented using recursive functions

12.7 Inorder Traversal of Trees

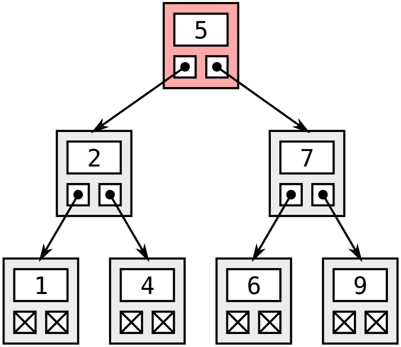
```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```



Output

12.7 Inorder Traversal of Trees

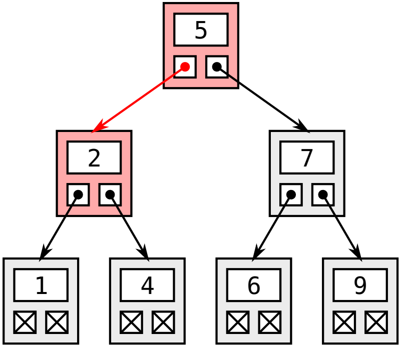
```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```



Output

12.7 Inorder Traversal of Trees

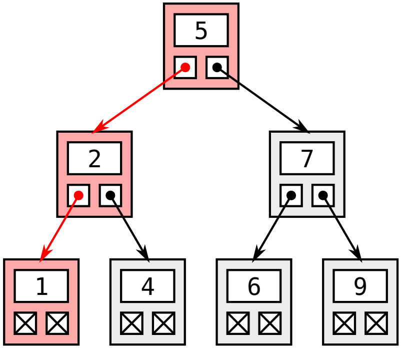
```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```



Output

12.7 Inorder Traversal of Trees

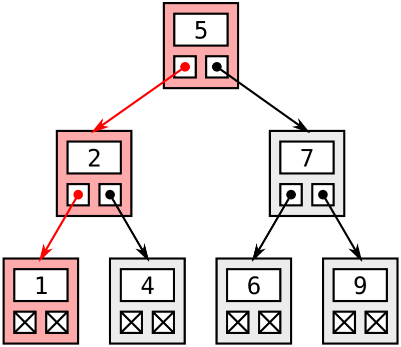
```c
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```



Output

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

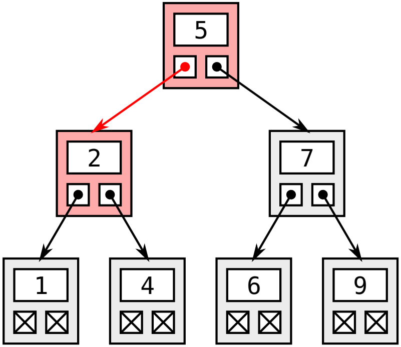


Output

```
  1
```

12.7 Inorder Traversal of Trees

```c
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

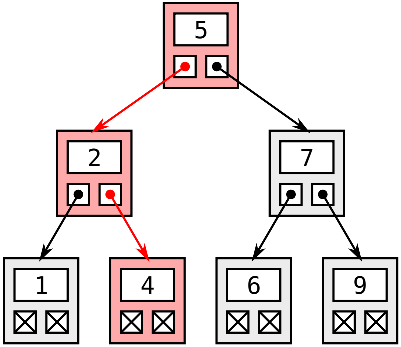


Output

```
1 2
```

12.7 Inorder Traversal of Trees

```c
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

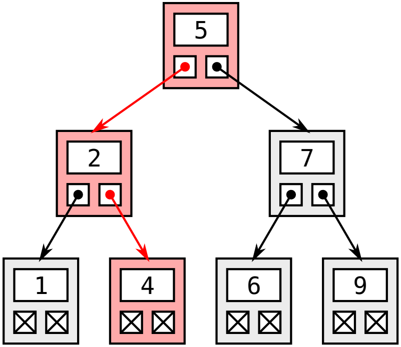


Output

```
1 2
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

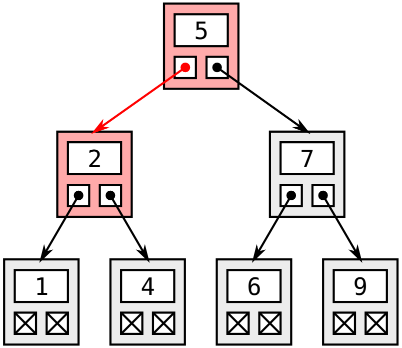


Output

```
1 2 4
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

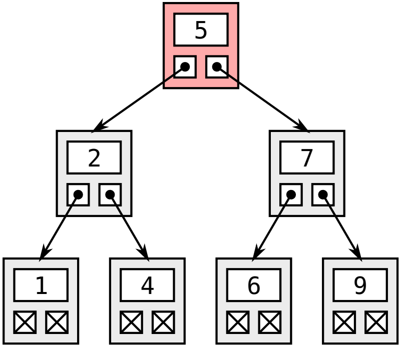


Output

```
 1 2 4
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

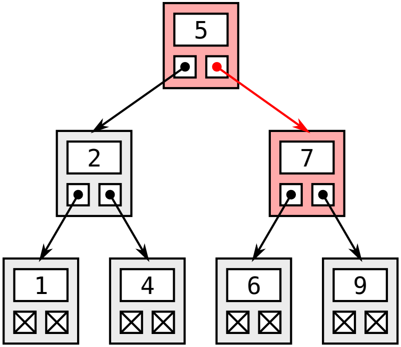


Output

```
 1  2  4  5
```

12.7 Inorder Traversal of Trees

```c
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

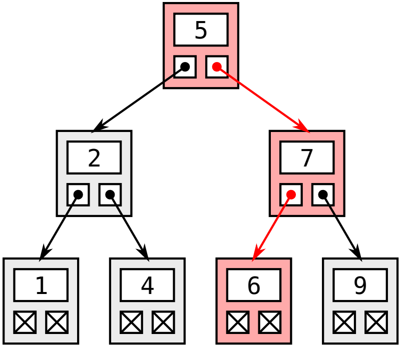


Output

```
 1 2 4 5
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

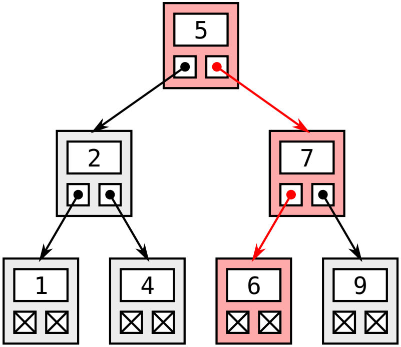


Output

```
1 2 4 5
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

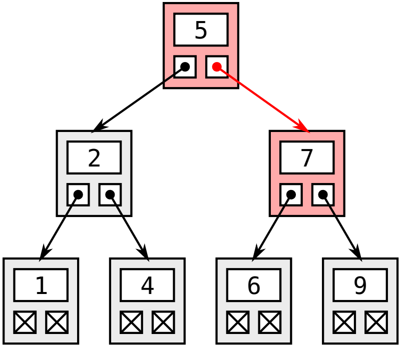


Output

```
 1  2  4  5  6
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

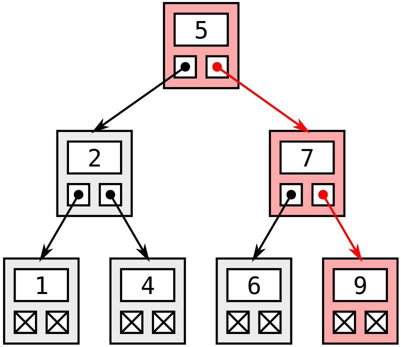


Output

```
1 2 4 5 6 7
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

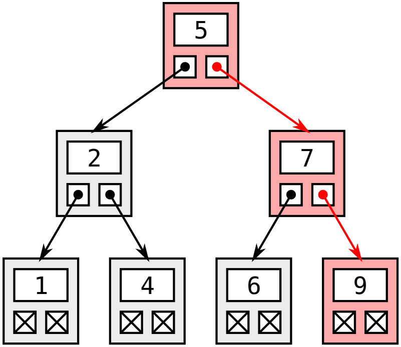


Output

```
1  2  4  5  6  7
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

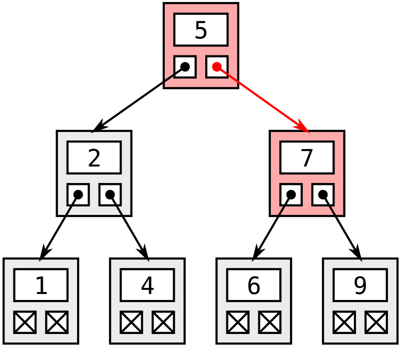


Output

```
1 2 4 5 6 7 9
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

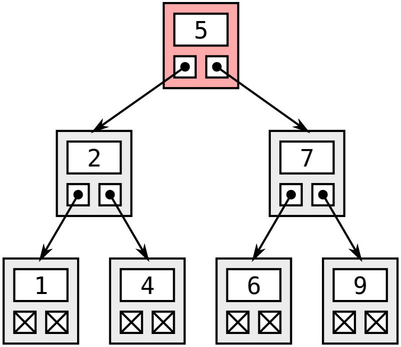


Output

```
1 2 4 5 6 7 9
```

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```

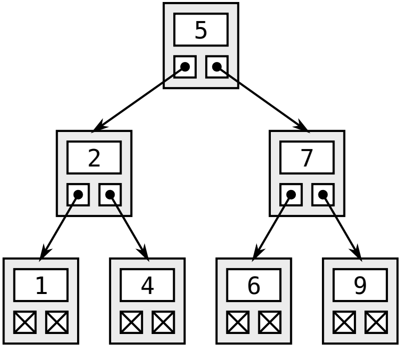


Output

```
1 2 4 5 6 7 9
```

12.7 Preorder Traversal of Trees

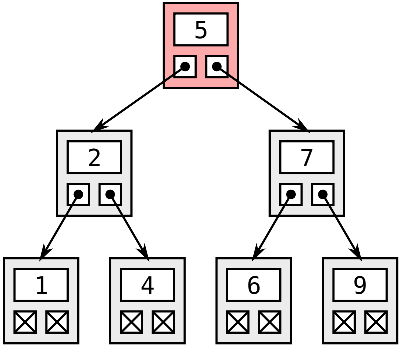
```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```



Output

12.7 Preorder Traversal of Trees

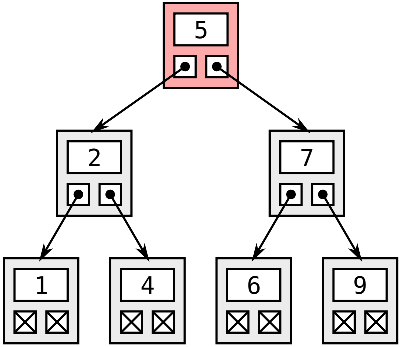
```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```



Output

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

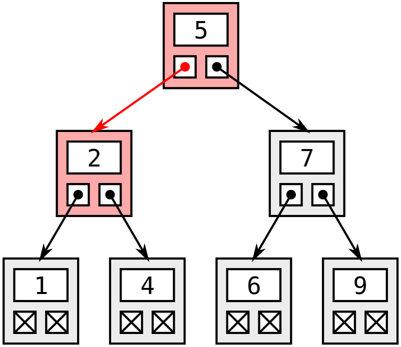


Output

```
5
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

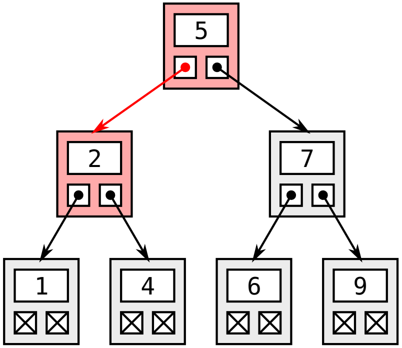


Output

```
5
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

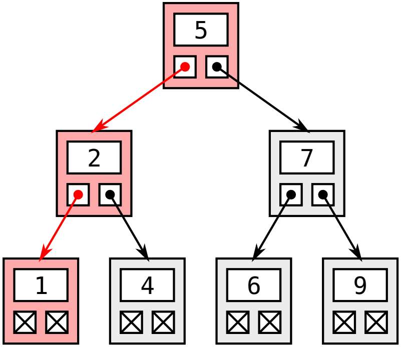


Output

```
5 2
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

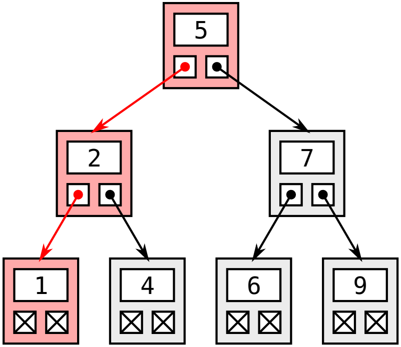


Output

```
5 2
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

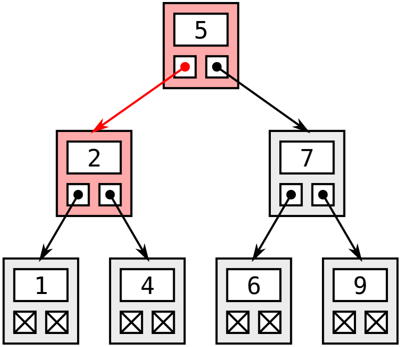


Output

```
5 2 1
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

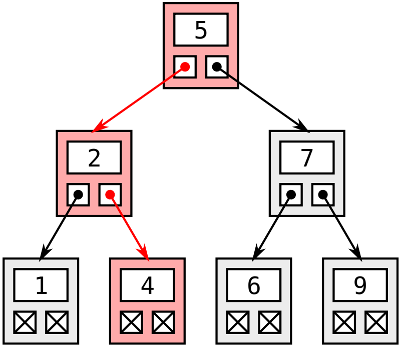


Output

```
 5 2 1
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

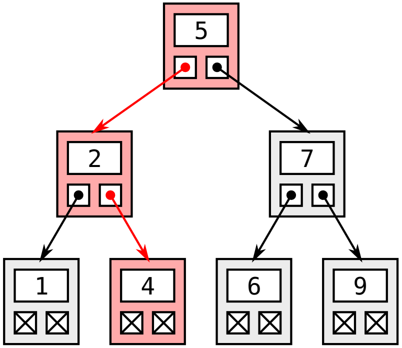


Output

```
 5 2 1
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

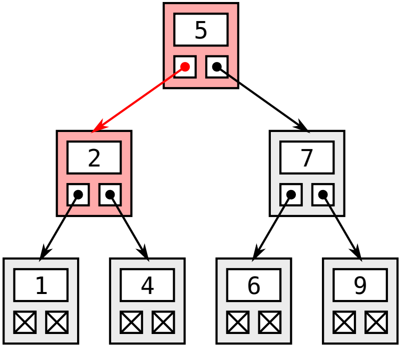


Output

```
5 2 1 4
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

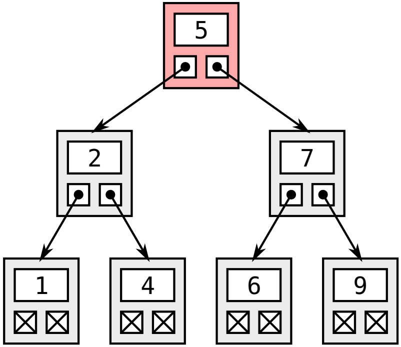


Output

```
 5 2 1 4
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

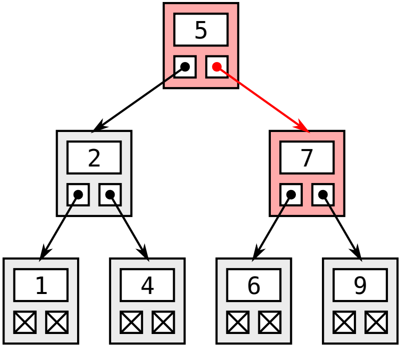


Output

```
5 2 1 4
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

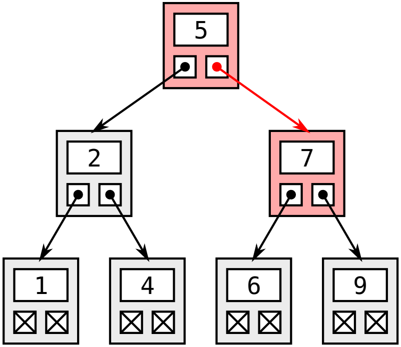


Output

```
5 2 1 4
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

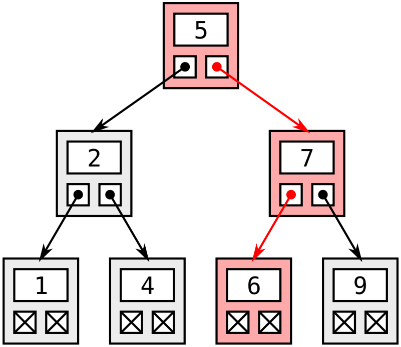


Output

```
5 2 1 4 7
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

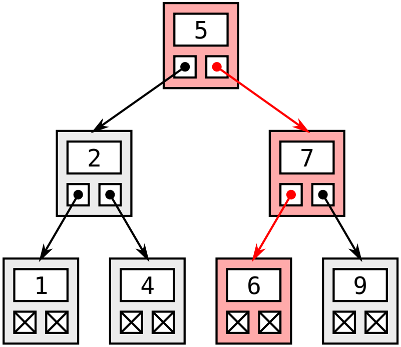


Output

```
5 2 1 4 7
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

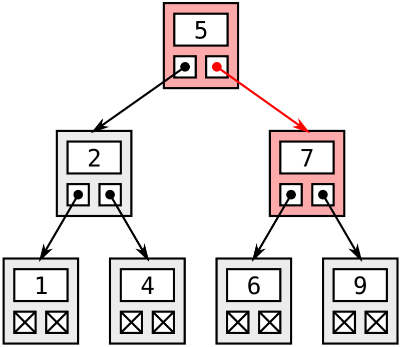


Output

```
5 2 1 4 7 6
```

12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

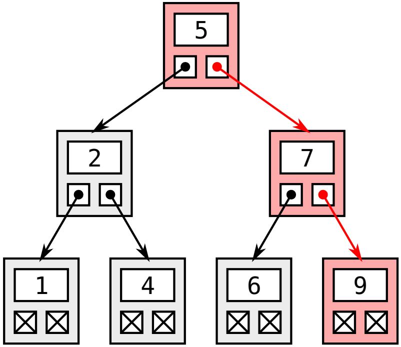


Output

```
5 2 1 4 7 6
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

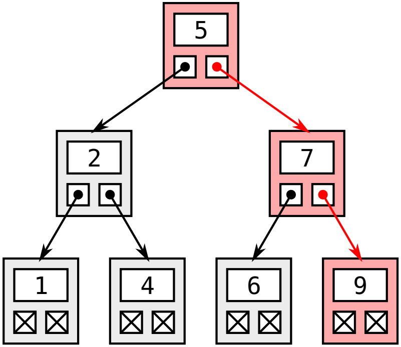


Output

```
 5 2 1 4 7 6
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

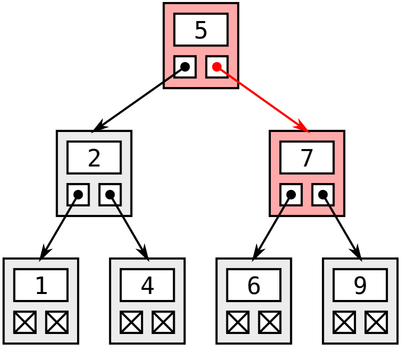


Output

```
5 2 1 4 7 6 9
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

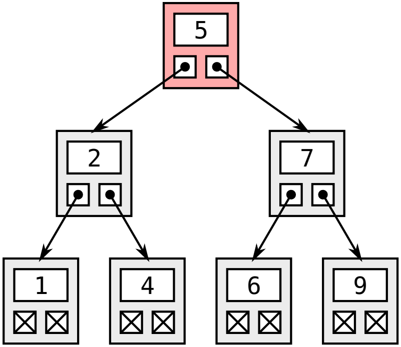


Output

```
5 2 1 4 7 6 9
```

12.7 Preorder Traversal of Trees

```c
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

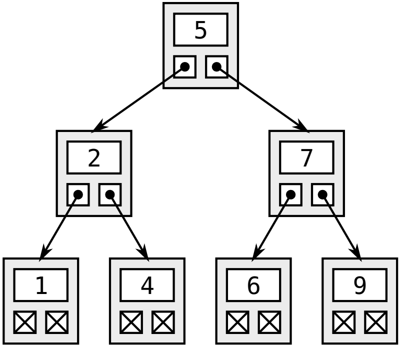


Output

```
5 2 1 4 7 6 9
```

12.7 Postorder Traversal of Trees

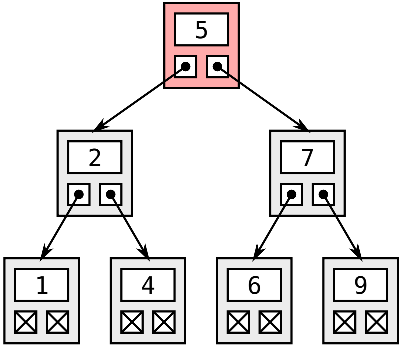
```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```



Output

12.7 Postorder Traversal of Trees

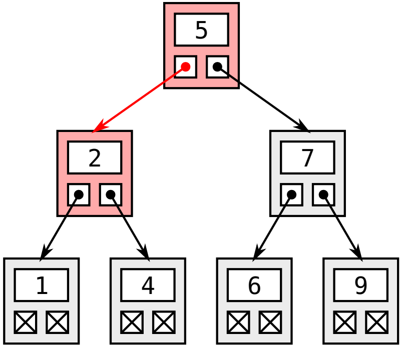
```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```



Output

12.7 Postorder Traversal of Trees

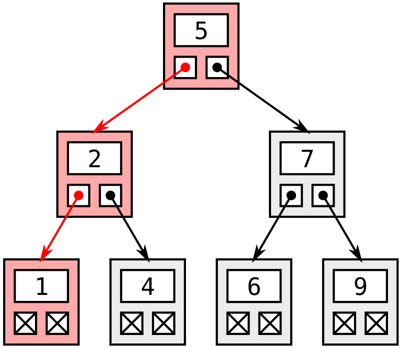
```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```



Output

12.7 Postorder Traversal of Trees

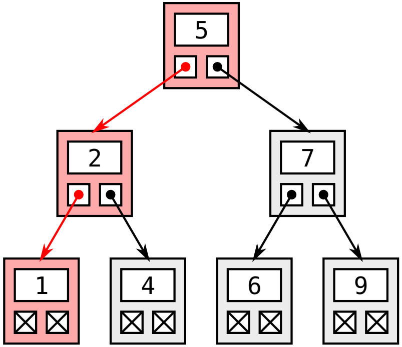
```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```



Output

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

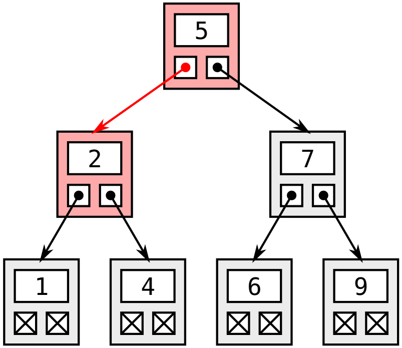


Output

```
  1
```

12.7 Postorder Traversal of Trees

```c
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

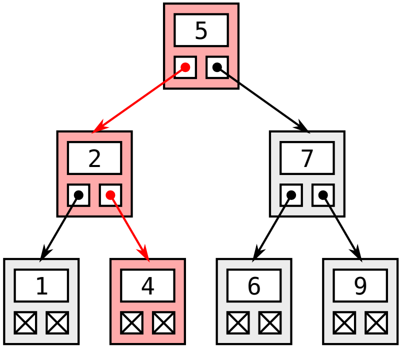


Output

```
1
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

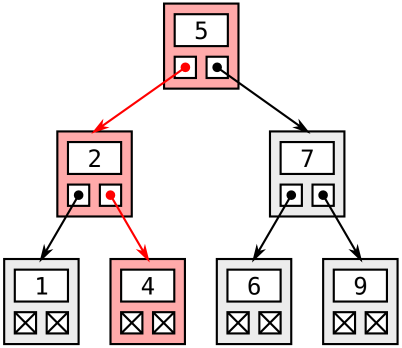


Output

```
1
```

12.7 Postorder Traversal of Trees

```c
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

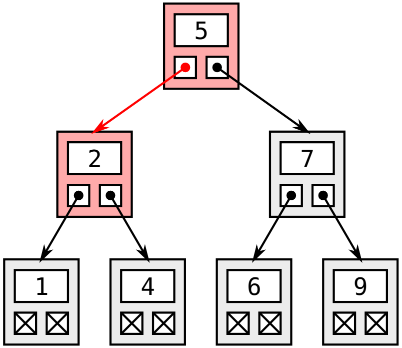


Output

```
 1  4
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

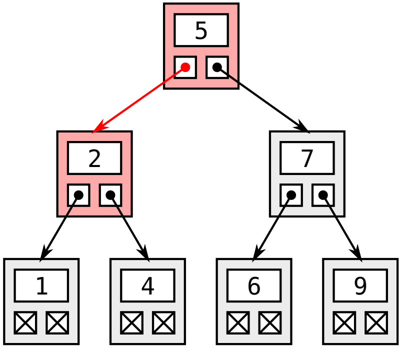


Output

```
 1 4
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

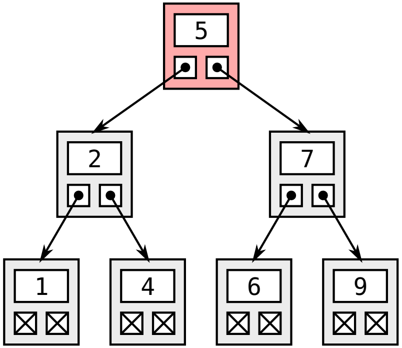


Output

```
1 4 2
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

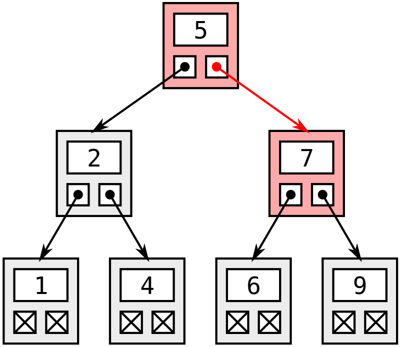


Output

```
1 4 2
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

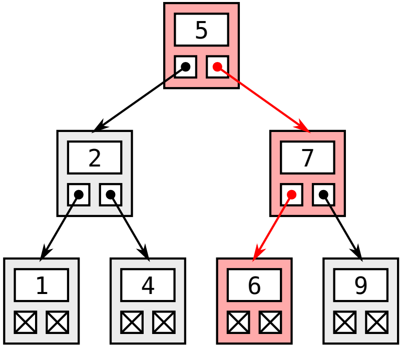


Output

```
 1 4 2
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

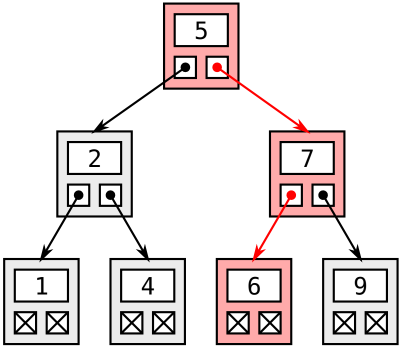


Output

```
 1  4  2
```

12.7 Postorder Traversal of Trees

```c
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

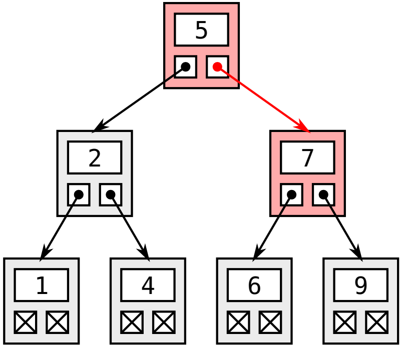


Output

```
1 4 2 6
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

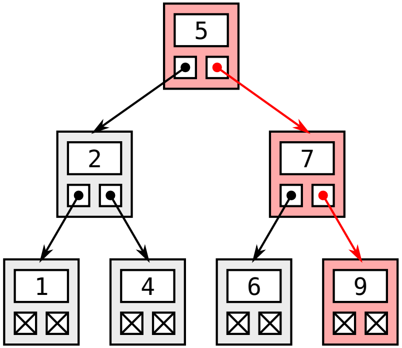


Output

```
1 4 2 6
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

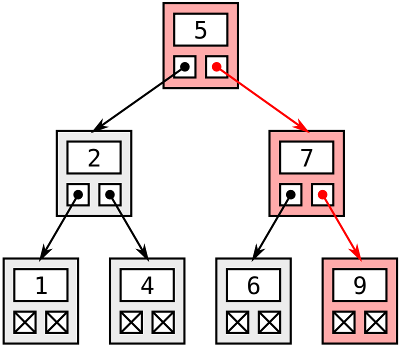


Output

```
1 4 2 6
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

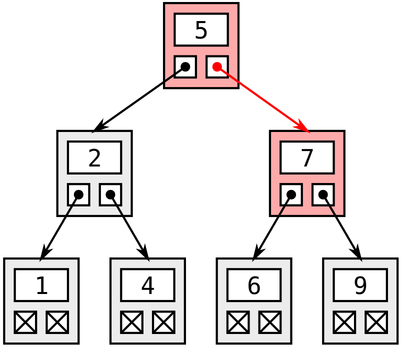


Output

```
1 4 2 6 9
```

12.7 Postorder Traversal of Trees

```c
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

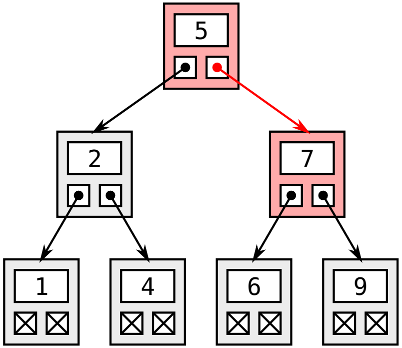


Output

```
1 4 2 6 9
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

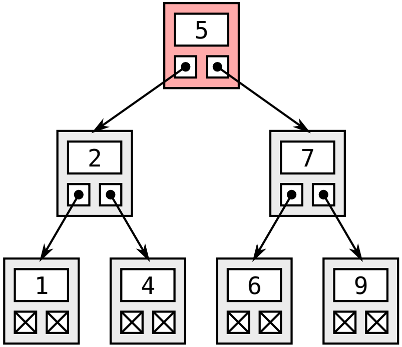


Output

```
1 4 2 6 9 7
```

12.7 Postorder Traversal of Trees

```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```

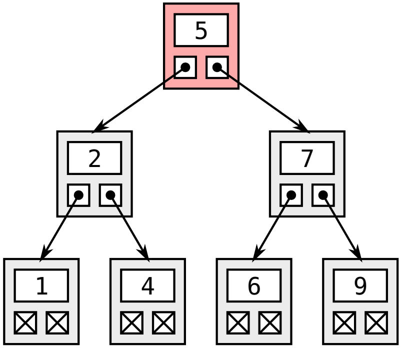


Output

```
1 4 2 6 9 7
```

12.7 Postorder Traversal of Trees

```c
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```



Output

```
1 4 2 6 9 7 5
```

Perspective

Today

Dynamic Data Structures III

- Trees

Next lecture

Object-Oriented Programming I

- Introduction to OOP

Homework

1. Study Section 12.7 in Deitel & Deitel
2. Do Self Review Exercises 12.1, 12.5 in Deitel & Deitel
3. Do Exercise 12.23 in Deitel & Deitel