

BINARY SEARCH TREES (CONTRD)

Problem Solving with Computers-II

C++

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

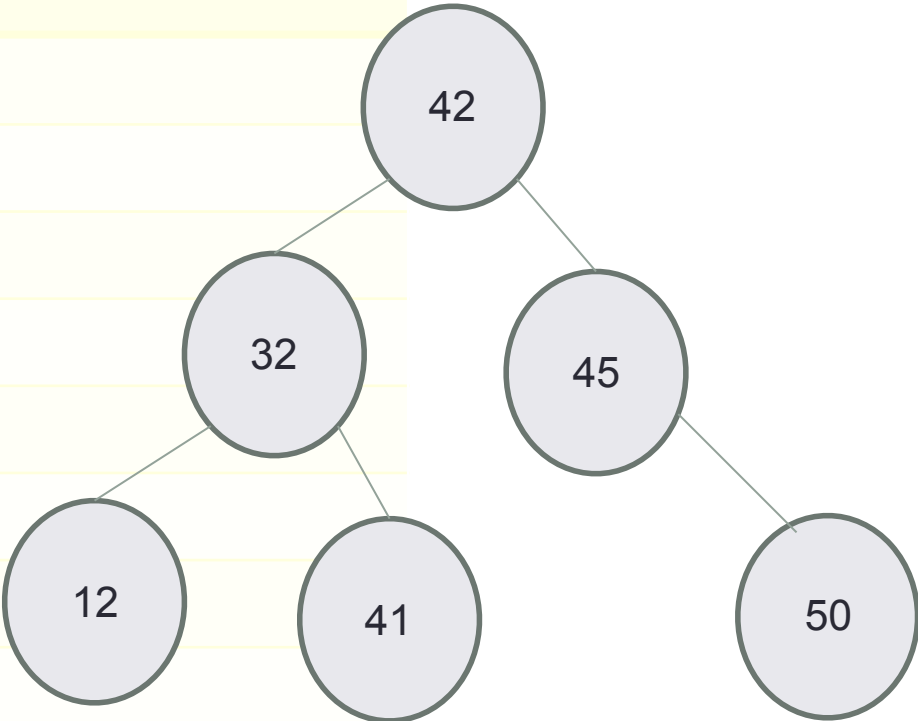
int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

A node in a BST

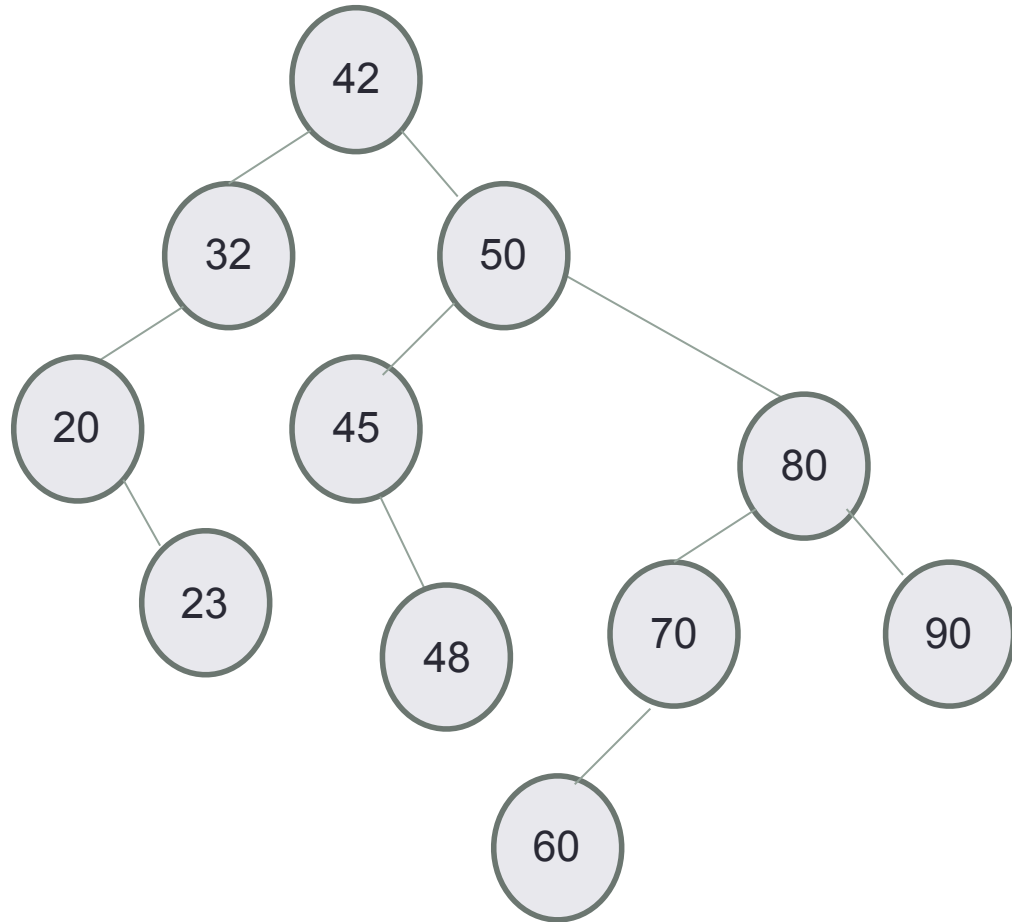
```
class BSTNode {  
  
public:  
    BSTNode* left;  
    BSTNode* right;  
    BSTNode* parent;  
    int const data;  
  
    BSTNode( const int & d ) : data(d) {  
        left = right = parent = 0;  
    }  
};
```

Define the BST ADT

Operations
Search
Insert
Min
Max
Successor
Predecessor
Delete
Print elements in order

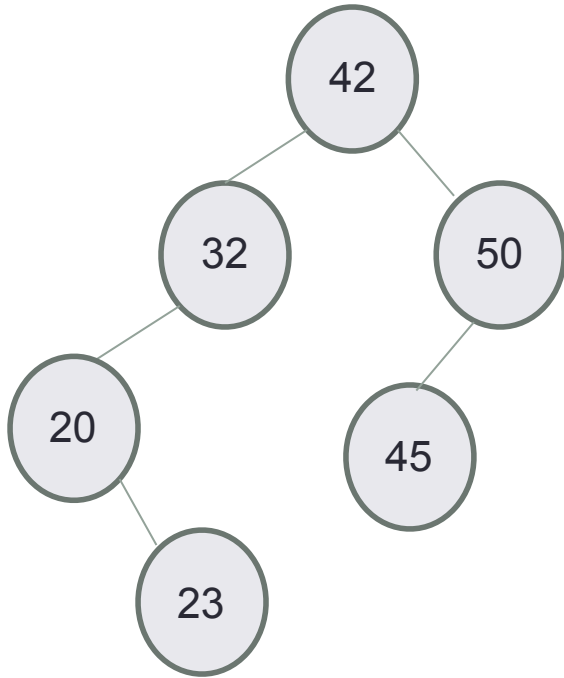


Successor: Next largest element



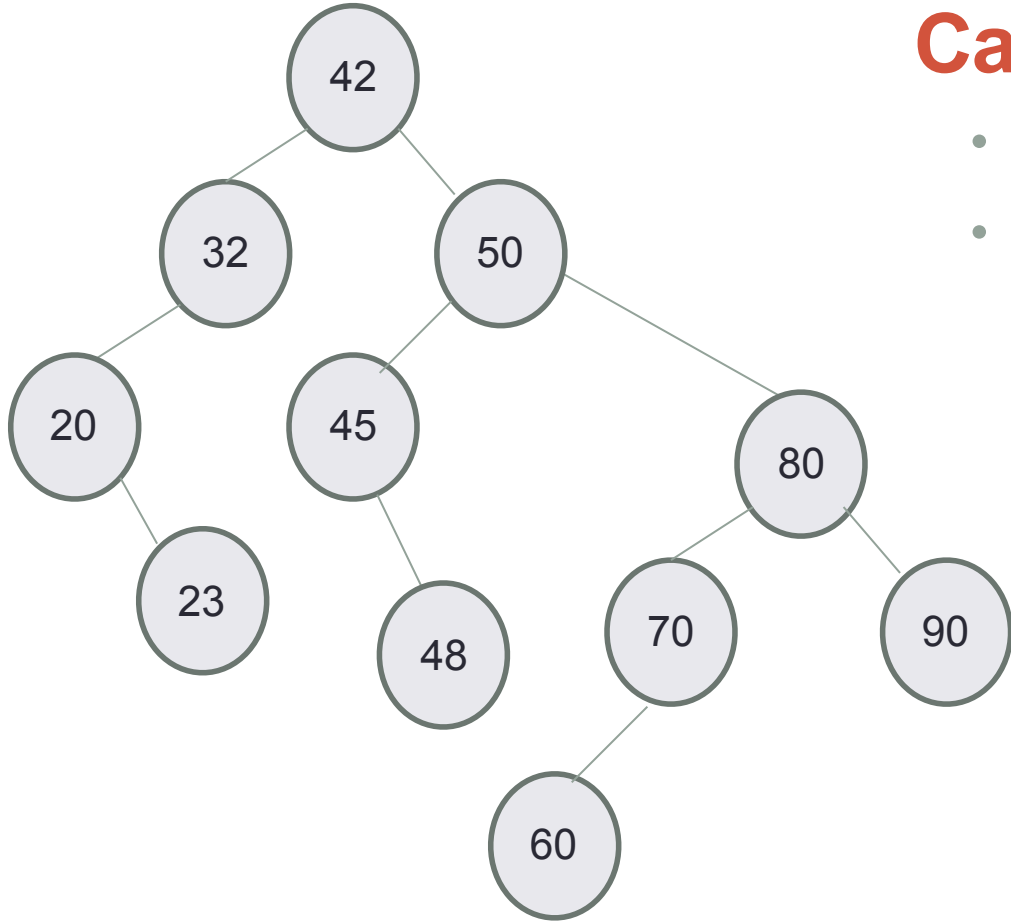
- What is the successor of 45?
- What is the successor of 50?
- What is the successor of 60?

Predecessor: Next smallest element



- What is the predecessor of 32?
- What is the predecessor of 45?

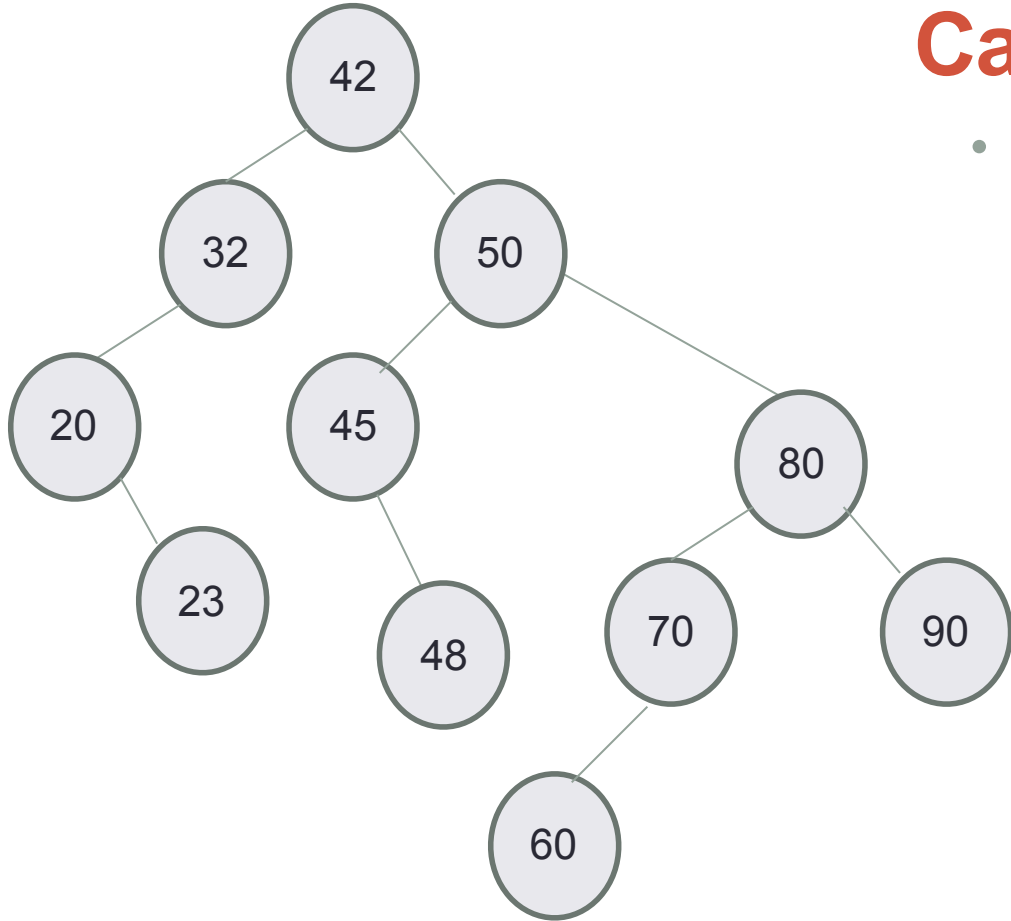
Delete: Case 1



Case 1: Node is a leaf node

- Set parent's (left/right) child pointer to null
- Delete the node

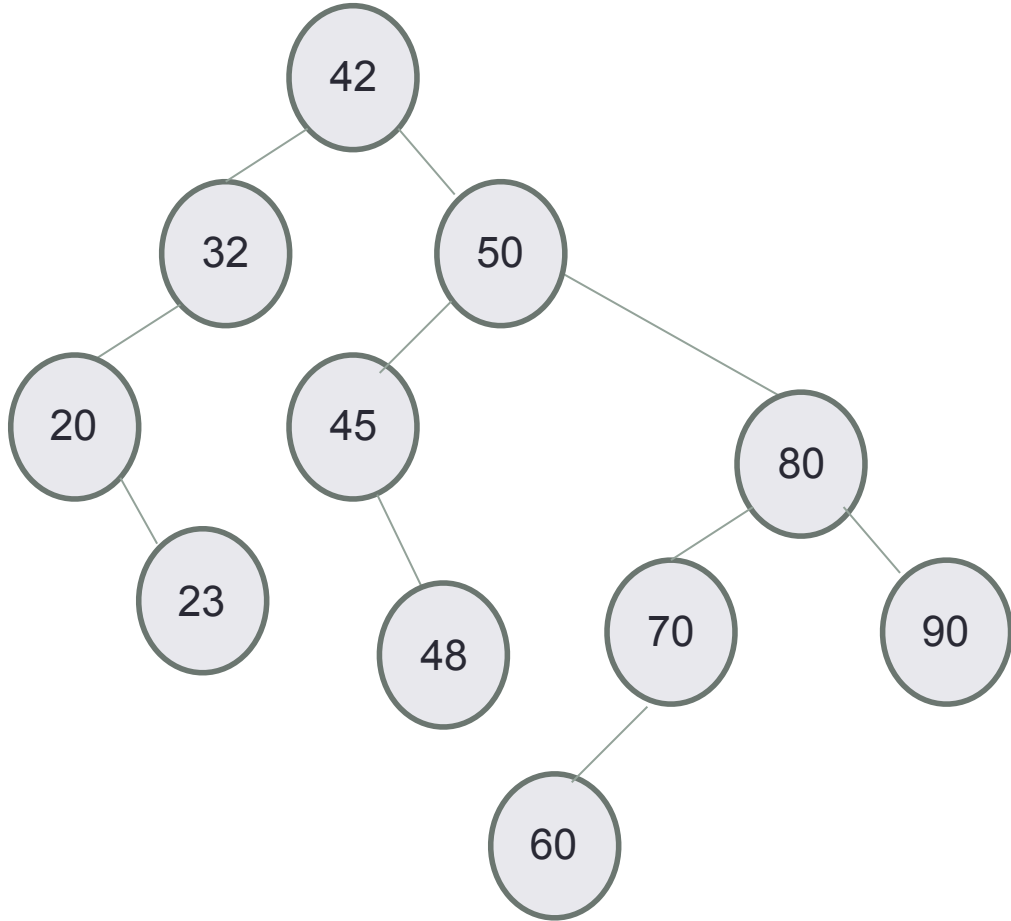
Delete: Case 2



Case 2 Node has only one child

- Replace the node by its only child

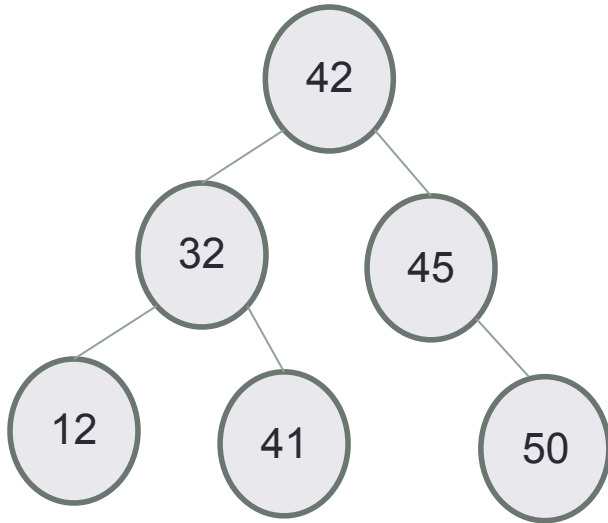
Delete: Case 3



Case 3 Node has two children

- Can we still replace the node by one of its children? Why or Why not?

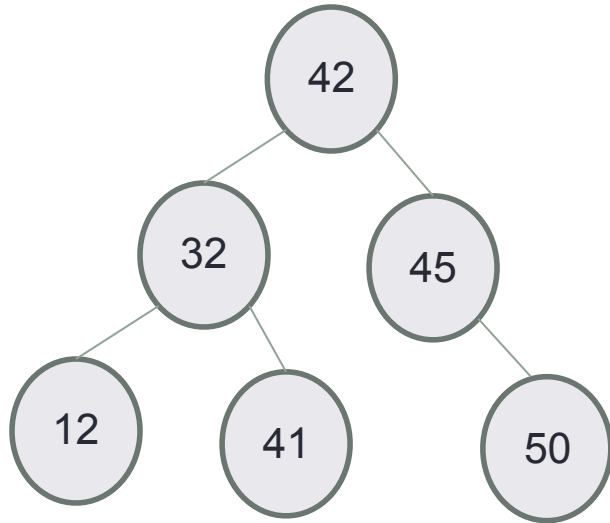
In order traversal: print elements in sorted order



Algorithm Inorder(tree)

1. Traverse the left subtree, i.e., call Inorder(left-subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right-subtree)

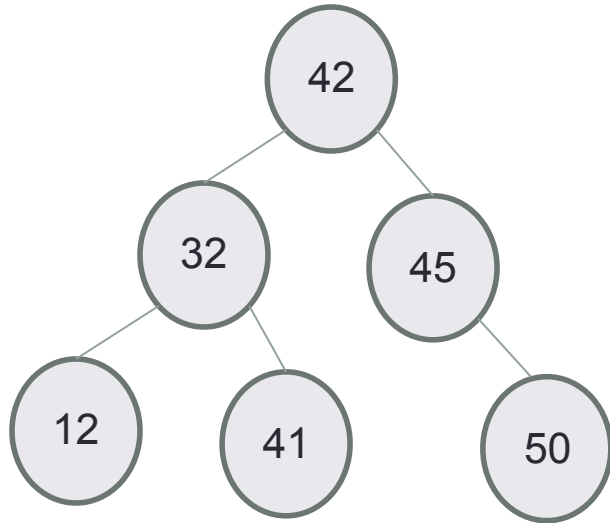
Pre-order traversal: nice way to linearize your tree!



Algorithm Preorder(tree)

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left-subtree)
3. Traverse the right subtree, i.e., call Preorder(right-subtree)

Post-order traversal: use in recursive destructors!



Algorithm Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left-subtree)
2. Traverse the right subtree, i.e., call Postorder(right-subtree)
3. Visit the root.

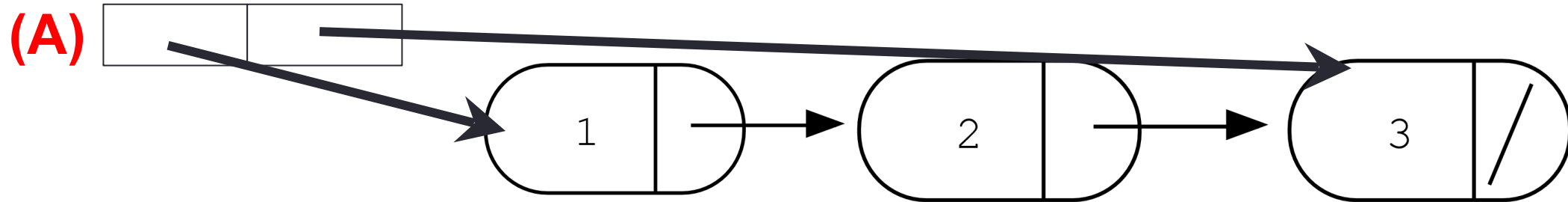
Concept Question

```
LinkedList::~~LinkedList(){  
    delete head;  
}
```

```
class Node {  
    public:  
        int info;  
        Node *next;  
};
```

Which of the following objects are deleted when the destructor of Linked-list is called?

head tail



(B): only the first node

(C): A and B

(D): All the nodes of the linked list

(E): A and D

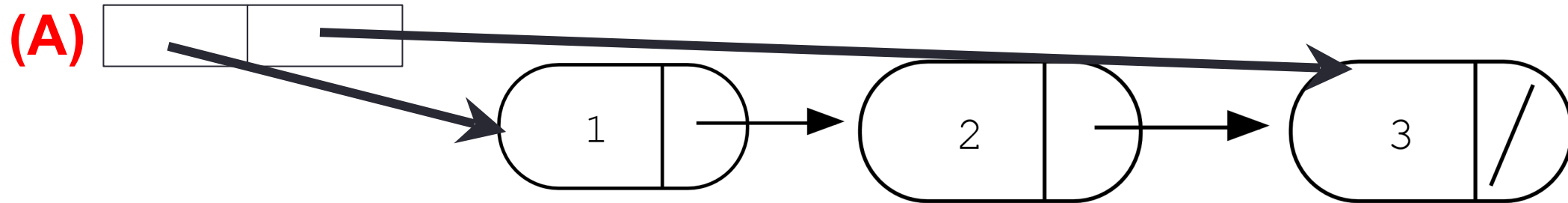
Concept Question

```
LinkedList::~~LinkedList(){  
    delete head;  
}
```

```
Node::~~Node(){  
    delete next;  
}
```

Which of the following objects are deleted when the destructor of Linked-list is called?

head tail



(B): All the nodes in the linked-list

(C): A and B

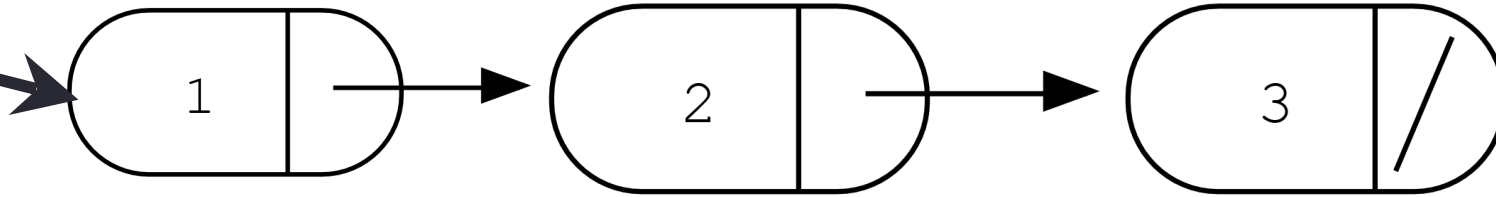
(D): Program crashes with a segmentation fault

(E): None of the above

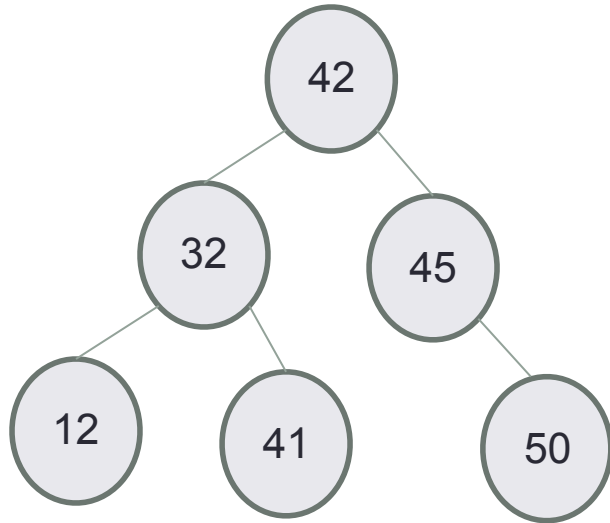
```
LinkedList::~~LinkedList(){  
    delete head;  
}
```

```
Node::~~Node(){  
    delete next;  
}
```

head tail



Post-order traversal: use in recursive destructors!



Algorithm Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left-subtree)
2. Traverse the right subtree, i.e., call Postorder(right-subtree)
3. Visit the root.