



DATA STRUCTURES AND ITS APPLICATIONS

UE19CS202

Shylaja S S & Kusuma K V

Department of Computer Science
& Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Binary Tree Traversal

Shylaja S S

Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Binary Tree Traversals



Important operation: Traversal

Traversal: Moving through all the nodes in a binary tree and visiting each one in turn

Trees: There are many orders possible since it is a nonlinear DS

- Tasks:
1. Visiting a node denoted by V
 2. Traversing the left subtree denoted by L
 3. Traversing the right subtree denoted by R

Six ways to arrange them: VLR, LVR, LRV, VRL, RVL, RLV

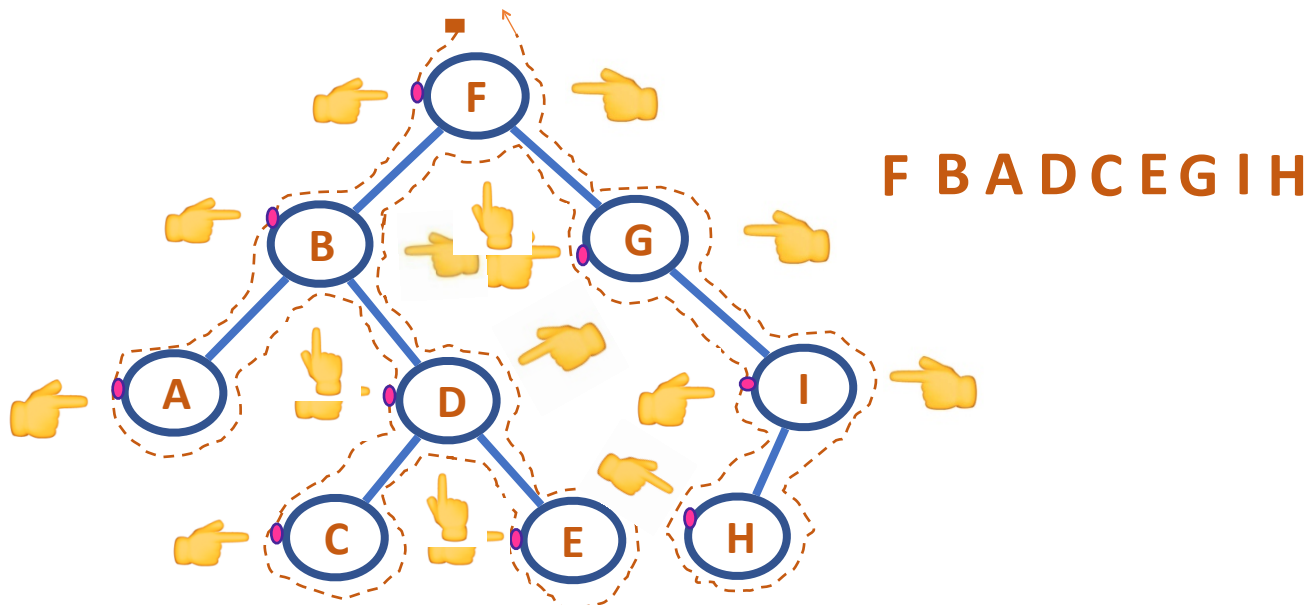
Standard Traversals include: VLR-Preorder, LVR-Inorder,
LRV-Postorder

DATA STRUCTURES AND ITS APPLICATIONS

Binary Tree Traversal: Preorder

Steps:

- Root Node is visited before the subtrees
- Left subtree is traversed in preorder
- Right subtree is traversed in preorder

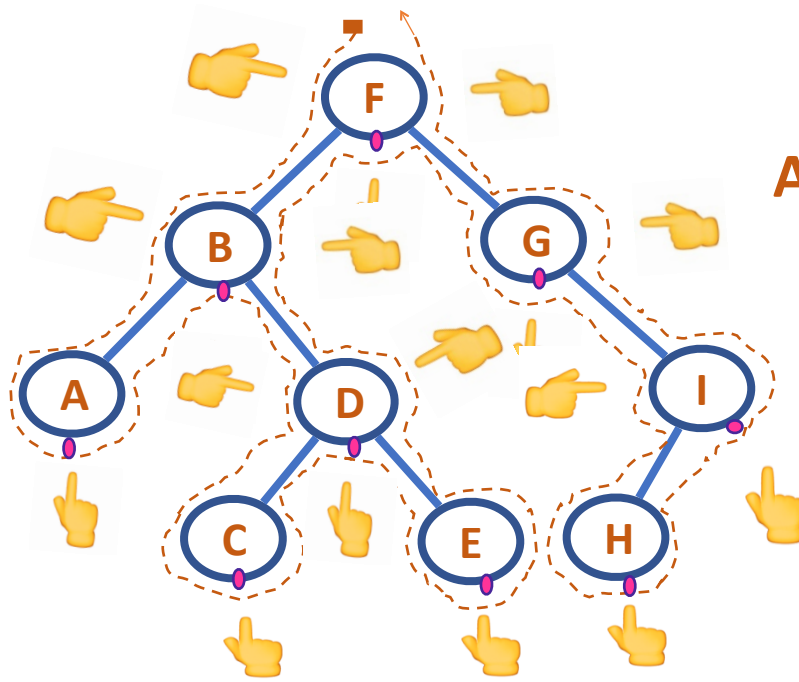


DATA STRUCTURES AND ITS APPLICATIONS

Binary Tree Traversal: Inorder

Steps:

- Left subtree is traversed in Inorder
- Root Node is visited
- Right subtree is traversed in Inorder



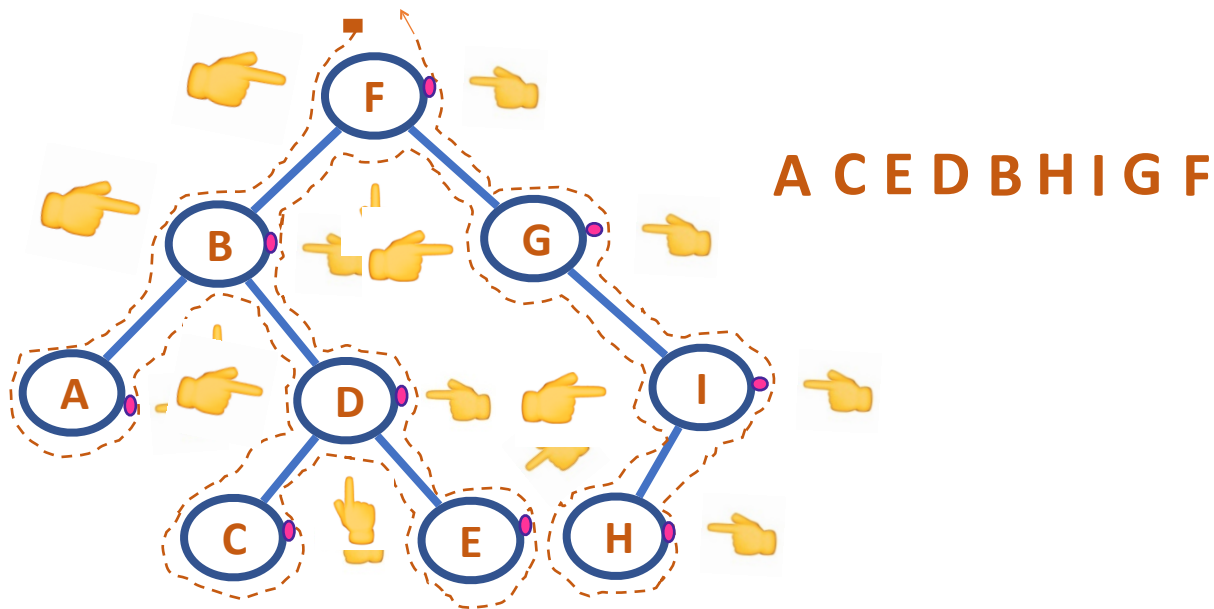
A B C D E F G H I

DATA STRUCTURES AND ITS APPLICATIONS

Binary Tree Traversal: Postorder

Steps:

- Left subtree is traversed in postorder
- Right subtree is traversed in postorder
- Root Node is visited



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal



```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null)
    {
        /* Travel down left branches as far as possible
        saving pointers to nodes passed in the stack*/
        push(s, current)
        current = current->left
    } //At this point, the left subtree is empty
    poppedNode = pop(s)
    print poppedNode ->info      //visit the node
    current = poppedNode ->right //traverse right subtree
} while(!isEmpty(s) or current != null)
```

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
```

```
  s = emptyStack ➡
```

```
  current = root ➡
```

```
  do { ➡
```

```
    while(current != null) ➡
```

```
    {
```

```
      push(s, current) ➡
```

```
      current = current->left
```

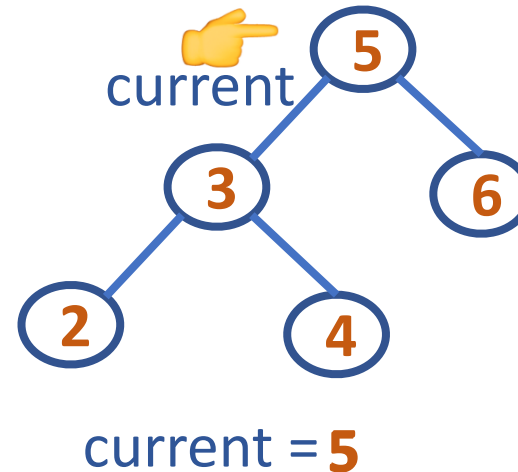
```
    }
```

```
    poppedNode = pop(s)
```

```
    print poppedNode ->info
```

```
    current = poppedNode ->right
```

```
  } while(!isEmpty(s) or current != null)
```

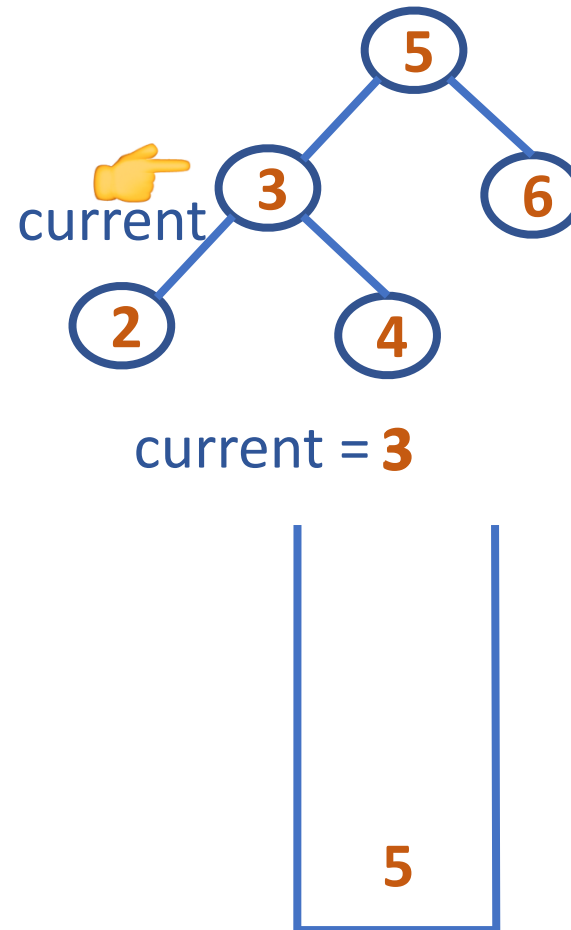


Note: Stack has Address of Nodes Pushed In

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null)
    {
        push(s, current)
        current = current->left
    }
    poppedNode = pop(s)
    print poppedNode ->info
    current = poppedNode ->right
} while(!isEmpty(s) or current != null)
```



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
```

```
  s = emptyStack
```

```
  current = root
```

```
  do {
```

```
    while(current != null)  ➡
```

```
    {
```

```
      push(s, current)  ➡
```

```
      current = current->left  ➡
```

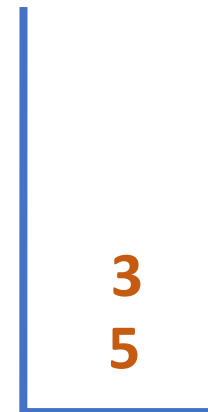
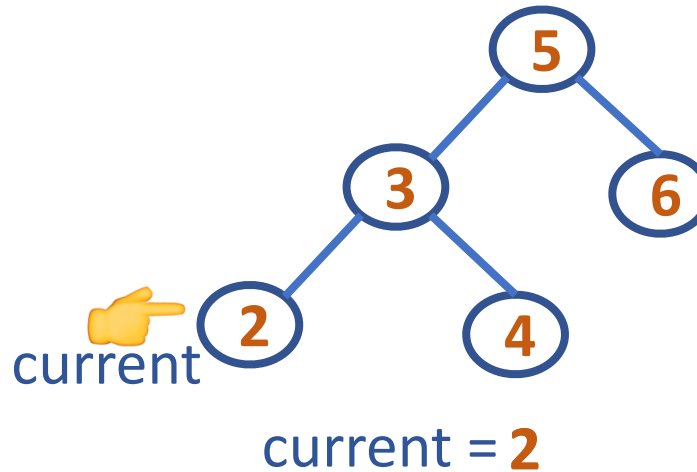
```
    }
```

```
    poppedNode = pop(s)
```

```
    print poppedNode ->info
```

```
    current = poppedNode ->right
```

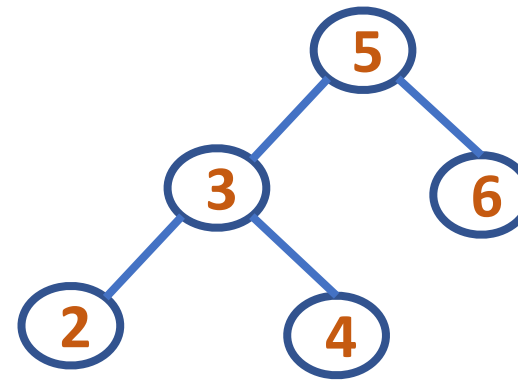
```
  } while(!isEmpty(s) or current != null)
```



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null)  ➡
    {
        push(s, current)
        current = current->left  ➡
    }
    poppedNode = pop(s)
    print poppedNode ->info
    current = poppedNode ->right
} while(!isEmpty(s) or current != null)
```



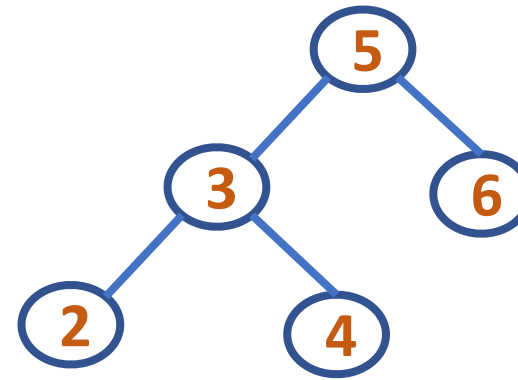
current = **N**

2
3
5

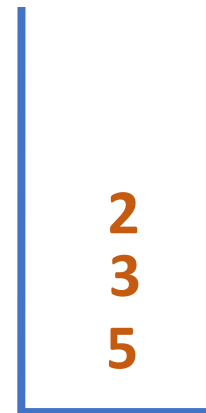
DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null)
    {
        push(s, current)
        current = current->left
    }
    poppedNode = pop(s) ➡ poppedNode =
    print poppedNode ->info ➡
    current = poppedNode ->right ➡
} while(!isEmpty(s) or current != null) ➡
```



current = **N**



Inorder Traversal:

2

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
```

```
  s = emptyStack
```

```
  current = root
```

```
  do {
```

```
    while(current != null)
```

```
    {
```

```
      push(s, current)
```

```
      current = current->left
```

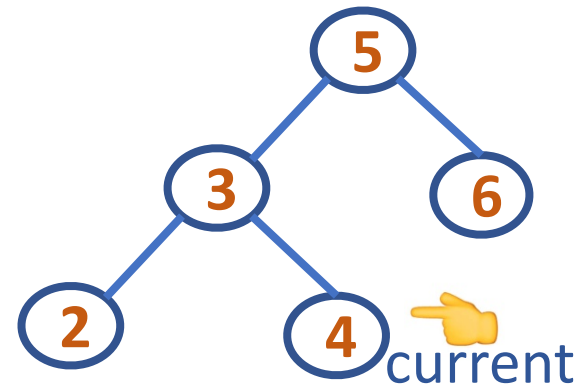
```
    }
```

```
    poppedNode = pop(s)
```

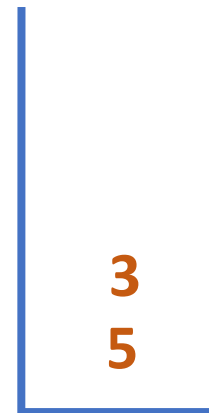
```
    print poppedNode ->info
```

```
    current = poppedNode ->right
```

```
  } while(!isEmpty(s) or current != null )
```



current = ~~4~~



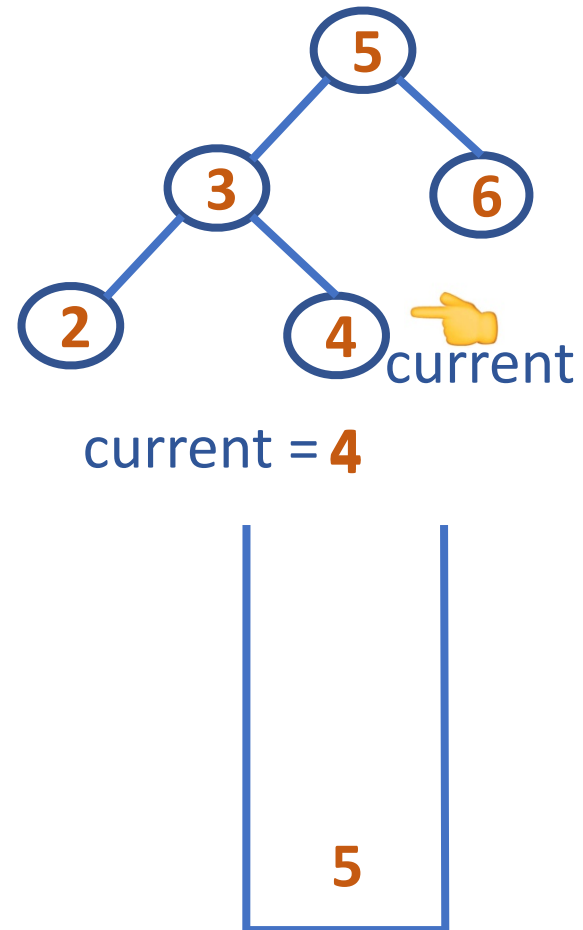
Inorder Traversal:

2 3

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null)
    {
        push(s, current)
        current = current->left
    }
    poppedNode = pop(s)
    print poppedNode ->info
    current = poppedNode ->right
} while(!isEmpty(s) or current != null )
```



Inorder Traversal:

2 3

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
```

```
  s = emptyStack
```

```
  current = root
```

```
  do {
```

```
    while(current != null) 🙌
```

```
    {
```

```
      push(s, current)
```

```
      current = current->left 🙌
```

```
    }
```

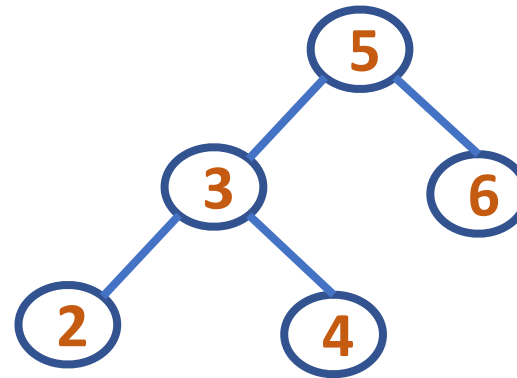
```
    poppedNode = pop(s) 🙌
```

poppedNode = 3

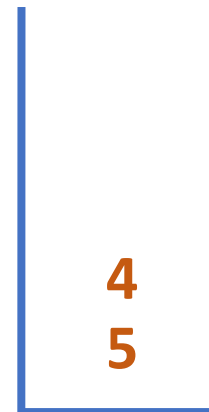
```
    print poppedNode ->info 🙌
```

```
    current = poppedNode ->right 🙌
```

```
  } while(!isEmpty(s) or current != null ) 🙌
```



current = N



Inorder Traversal:

2 3 4



PES
UNIVERSITY
ONLINE

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
```

```
  s = emptyStack
```

```
  current = root
```

```
  do {
```

```
    while(current != null)
```

```
    {
```

```
      push(s, current)
```

```
      current = current->left
```

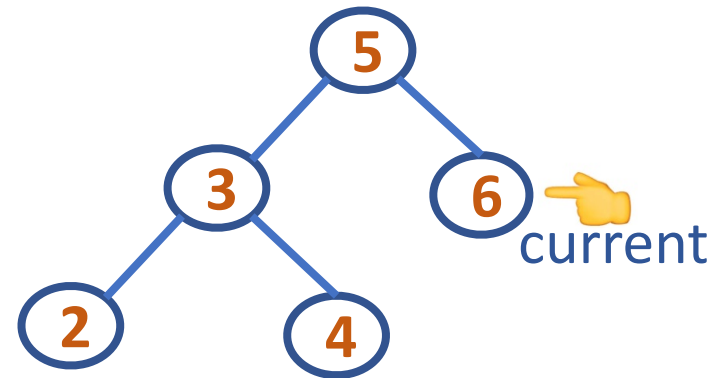
```
    }
```

```
    poppedNode = pop(s)
```

```
    print poppedNode ->info
```

```
    current = poppedNode ->right
```

```
  } while(!isEmpty(s) or current != null )
```



current = null



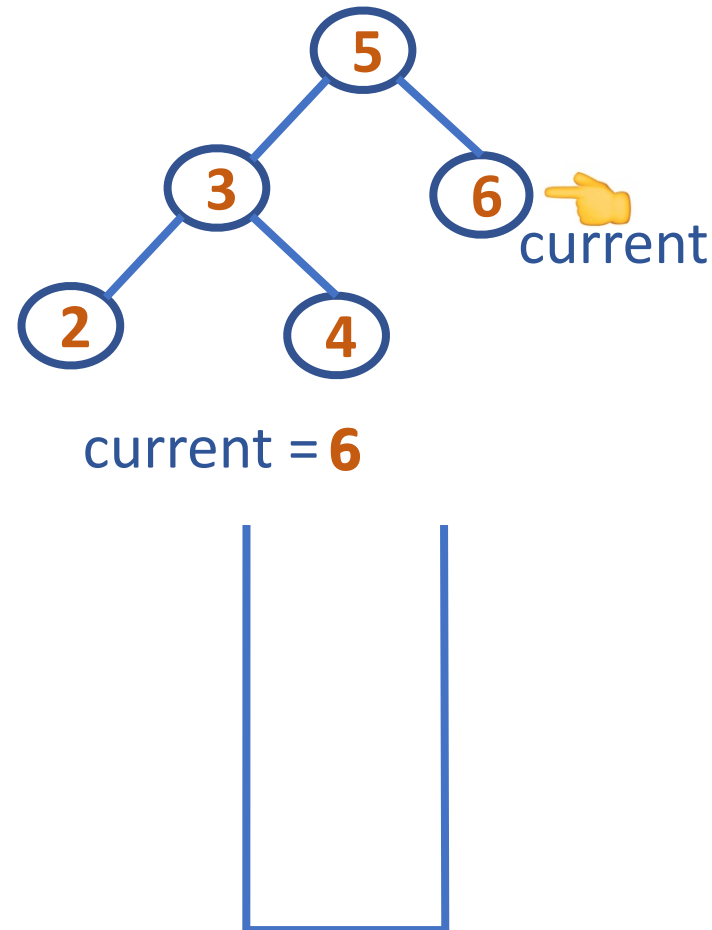
Inorder Traversal:

2 3 4 5

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null)
    {
        push(s, current)
        current = current->left
    }
    poppedNode = pop(s)
    print poppedNode ->info
    current = poppedNode ->right
} while(!isEmpty(s) or current != null )
```



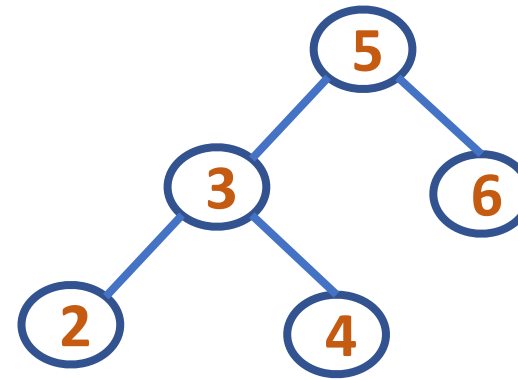
Inorder Traversal:

2 3 4 5

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Inorder Traversal

```
iterativeInorder(root)
s = emptyStack
current = root
do {
    while(current != null) 📌
    {
        push(s, current)
        current = current->left 📌
    }
    poppedNode = pop(s) 📌
    print poppedNode ->info 📌
    current = poppedNode ->right 📌
} while(!isEmpty(s) or current != null) 📌
```



current = **N**



Inorder Traversal:

2 3 4 5 6

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

```
iterativePreorder(root)
current=root
if (current == null)
    return
s = emptyStack
push(s, current)
while(!isEmpty(s)) {
    current = pop(s)
    print current->info
    //right child is pushed first so that left is processed first
    if(current->right !=NULL)
        push(s, current->right)
    if(current->left !=NULL)
        push(s, current->left)
}
```

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

current=root ➡

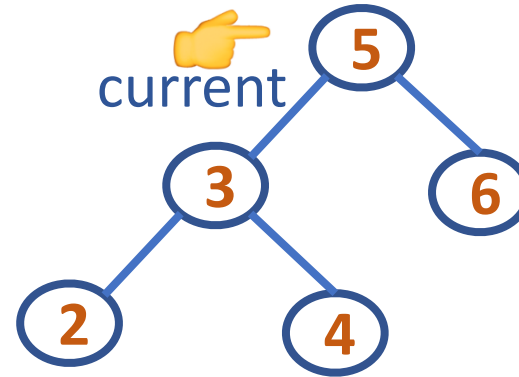
if (current == null) ➡

return

s = emptyStack ➡

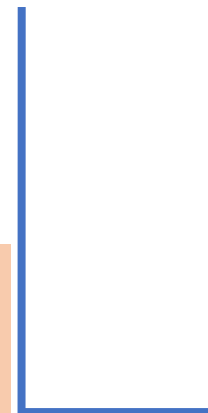
push(s, current) ➡

⋮



current = 5

Note: Stack has Address
of Nodes Pushed In



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

⋮

while (!isEmpty(s))
{

current = pop(s)

print current ->info

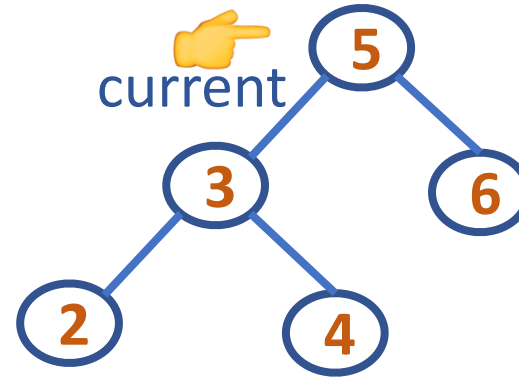
if(current->right != null)

push(s, current->right)

if(current->left != null)

push(s, current->left)

}



current = 5



Preorder Traversal:

5

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

```
iterativePreorder(root)
```

```
⋮
```

```
while (!isEmpty(s))
```

```
{
```

```
    current = pop(s)
```

```
    print current ->info
```

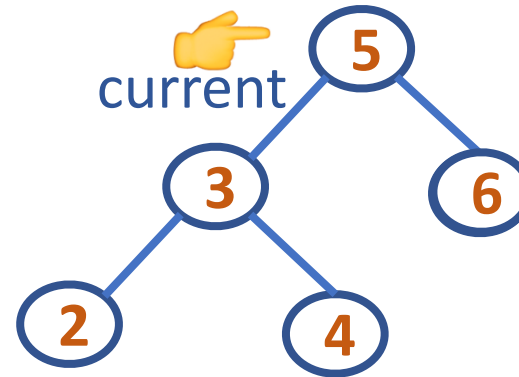
```
    if(current->right != null) ➡
```

```
        push(s, current->right) ➡
```

```
    if(current->left != null)
```

```
        push(s, current->left)
```

```
}
```



current = 5

current->right = 6



Preorder Traversal:

5

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

```
iterativePreorder(root)
```

```
    ⋮
```

```
while (!isEmpty(s))
```

```
{
```

```
    current = pop(s)
```

```
    print current ->info
```

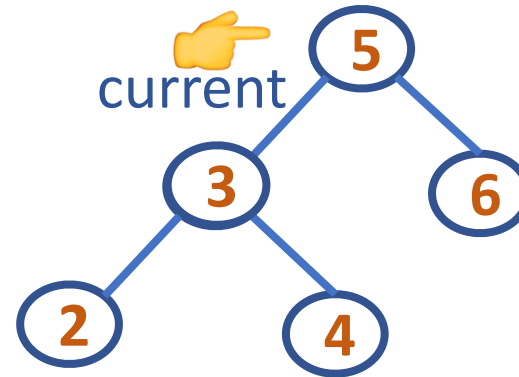
```
    if(current->right != null)
```

```
        push(s, current->right)
```

```
    if(current->left != null) ➡
```

```
        push(s, current->left) ➡
```

```
}
```



current = 5

current->left = 3



Preorder Traversal:

5

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

⋮

while (!isEmpty(s))



{

current = pop(s)



print current ->info



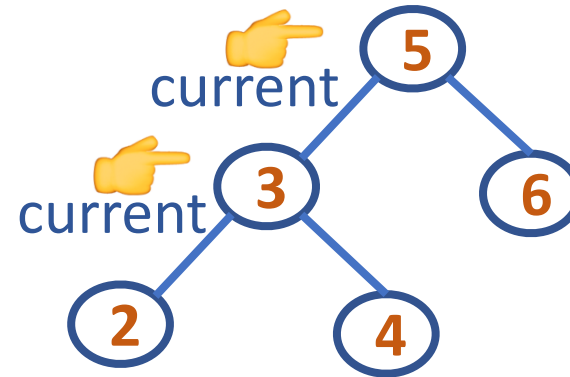
if(current->right != null)

push(s, current->right)

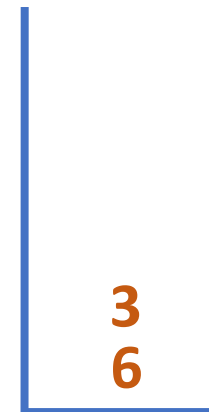
if(current->left != null)

push(s, current->left)

}



current = 3



Preorder Traversal:

5 3

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

⋮

while (!isEmpty(s))

{

current = pop(s)

print current ->info

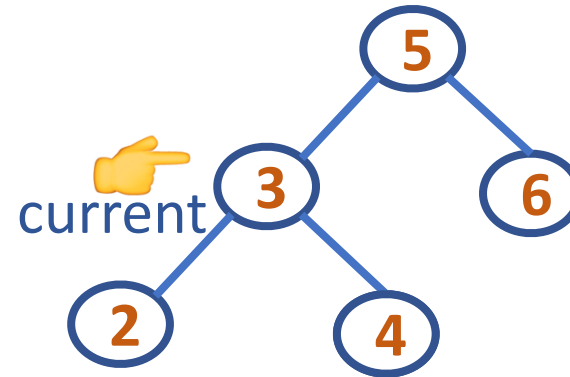
if(current->right != null) ➡

push(s, current->right) ➡

if(current->left != null)

push(s, current->left)

}



current = 3

current->right = 4



Preorder Traversal:

5 3

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

⋮

while (!isEmpty(s))

{

current = pop(s)

print current ->info

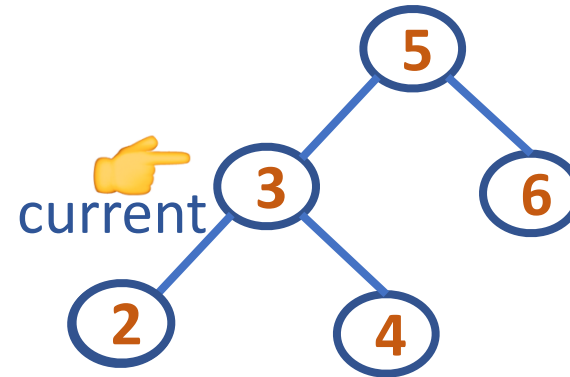
if(current->right != null)

push(s, current->right)

if(current->left != null) ➡

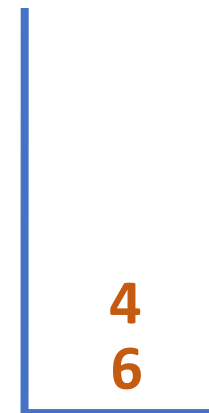
push(s, current->left) ➡

}



current = 3

current->left = 2



Preorder Traversal:

5 3

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

⋮

while (!isEmpty(s)) 📌
{

current = pop(s) 📌

print current ->info 📌

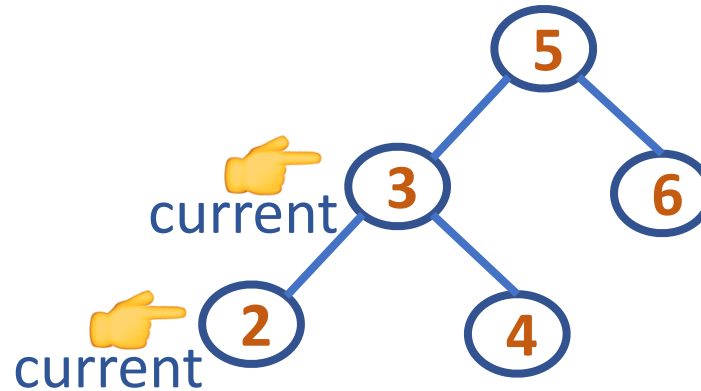
if(current->right != null)

push(s, current->right)

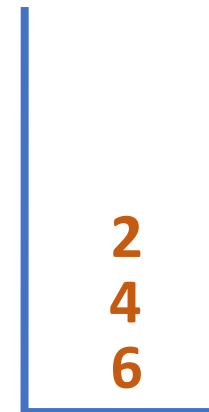
if(current->left != null)

push(s, current->left)

}



current = 3



Preorder Traversal:

5 3 2

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)



```
while (!isEmpty(s))
```

```
{
```

```
    current = pop(s)
```

```
    print current ->info
```

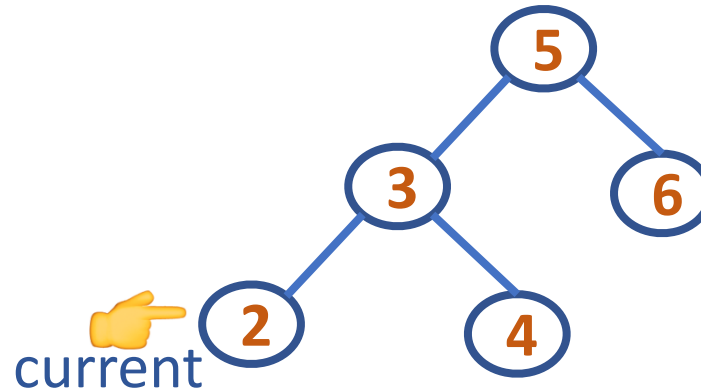
```
    if(current->right != null) ➡
```

```
        push(s, current->right)
```

```
    if(current->left != null)
```

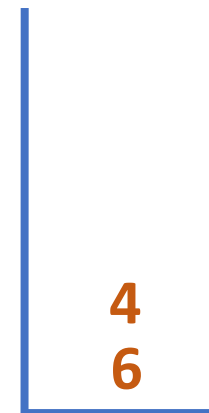
```
        push(s, current->left)
```

```
}
```



current = 2

current->right = N



PES
UNIVERSITY
ONLINE

Preorder Traversal:

5 3 2

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)



```
while (!isEmpty(s))
```

```
{
```

```
    current = pop(s)
```

```
    print current ->info
```

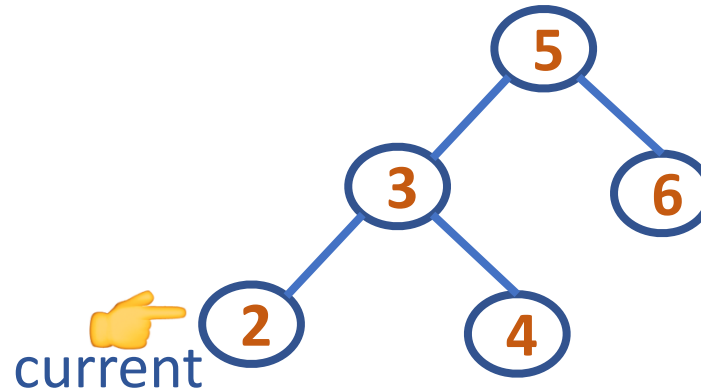
```
    if(current->right != null)
```

```
        push(s, current->right)
```

```
    if(current->left != null) ➡
```

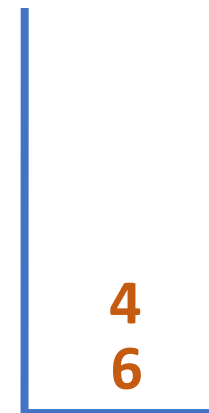
```
        push(s, current->left)
```

```
}
```



current = 2

current->left = N



PES
UNIVERSITY
ONLINE

Preorder Traversal:

5 3 2

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

```
iterativePreorder(root)
```

```
    ⋮
```

```
while (!isEmpty(s)) ➡
```

```
{
```

```
    current = pop(s) ➡
```

```
    print current ->info ➡
```

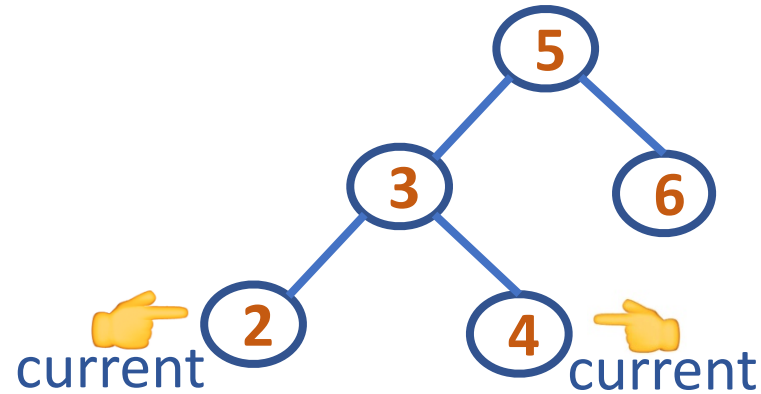
```
    if(current->right != null)
```

```
        push(s, current->right)
```

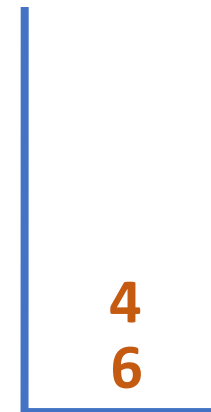
```
    if(current->left != null)
```

```
        push(s, current->left)
```

```
}
```



current = 2



Preorder Traversal:

5 3 2 4

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

```
iterativePreorder(root)
```



```
while (!isEmpty(s))
```

```
{
```

```
    current = pop(s)
```

```
    print current ->info
```

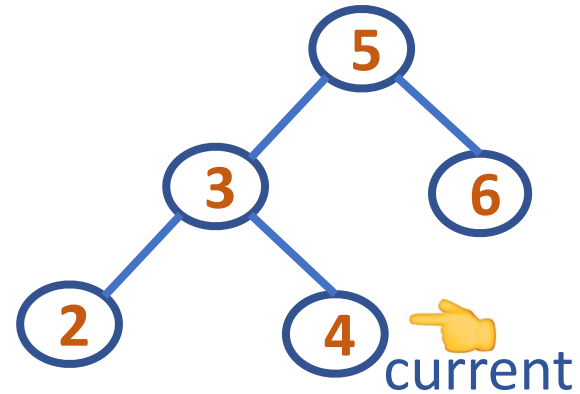
```
    if(current->right != null) 🖐️
```

```
        push(s, current->right)
```

```
    if(current->left != null) 🖐️
```

```
        push(s, current->left)
```

```
}
```



current = 4

current->right = N

current->left = N



Preorder Traversal:

5 3 2 4

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Preorder Traversal

iterativePreorder(root)

⋮

while (!isEmpty(s))
{

current = pop(s)

print current ->info

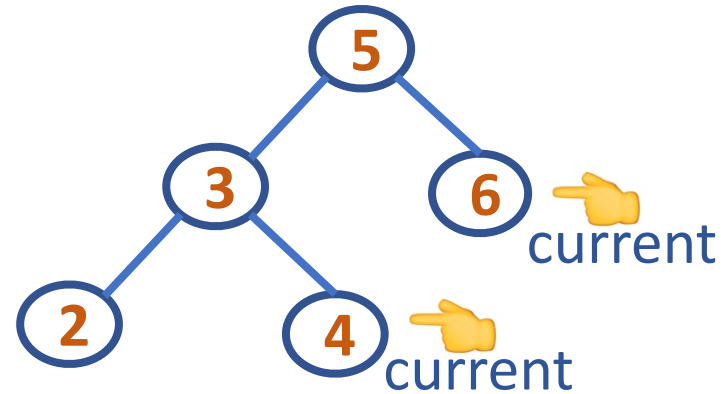
if(current->right != null)

push(s, current->right)

if(current->left != null)

push(s, current->left)

}



Preorder Traversal:

5 3 2 4 6

current = 4

current->right = N

current->left = N



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal



```
iterativePostorder(root)
s1 = emptyStack ; s2 = emptyStack ; push(s1, root)
while(!isEmpty(s1)) {
    current = pop(s1)
    push(s2,current)
    if(current->left !=NULL)
        push(s1, current->left)
    if(current->right !=NULL)
        push(s1, current->right)
}
while(!isEmpty(s2)) { //Print all the elements of stack2
    current = pop(s2)
    print current->info
}
```

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

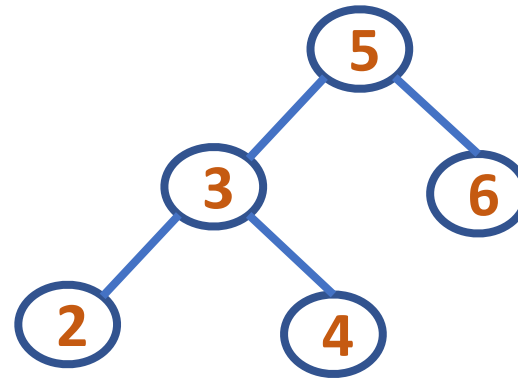
iterativePostorder(root)

s1 = emptyStack ➡

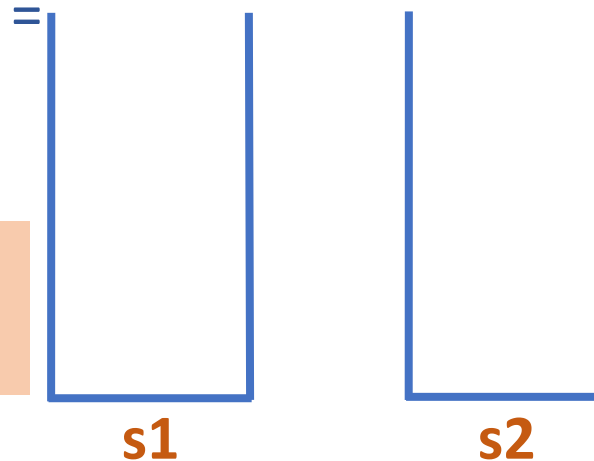
s2 = emptyStack ➡

push(s1, root) ➡

⋮



root 5



Note: Stacks have Address of Nodes Pushed In

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
    while(!isEmpty(s1)) ➡
```

```
    {
```

```
        current = pop(s1) ➡
```

```
        push(s2,current)
```

```
        if(current->left !=NULL)
```

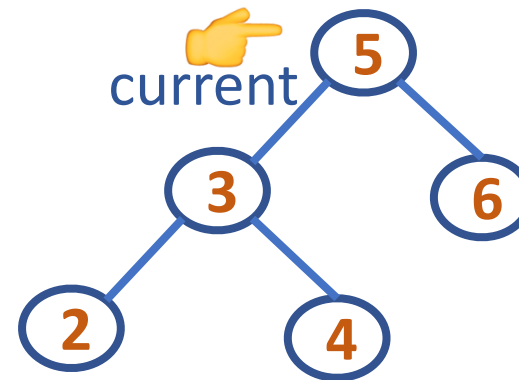
```
            push(s1, current->left)
```

```
        if(current->right !=NULL)
```

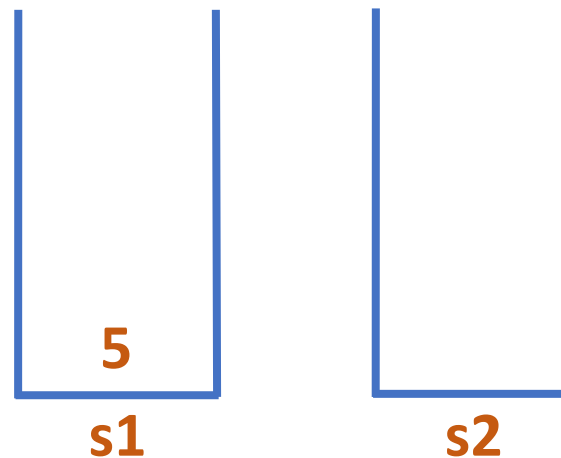
```
            push(s1, current->right)
```

```
    }
```

```
    ⋮
```



current =



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current) ➡
```

```
    if(current->left != NULL) ➡
```

```
        push(s1, current->left) ➡
```

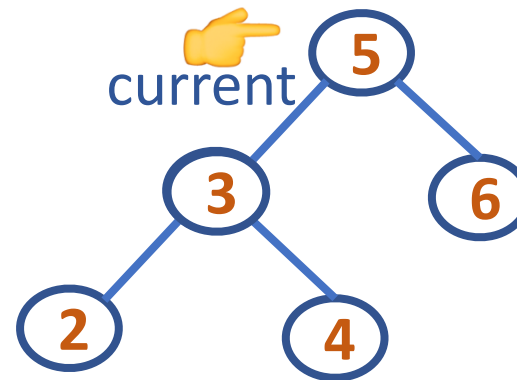
```
    if(current->right != NULL)
```

```
        push(s1, current->right)
```

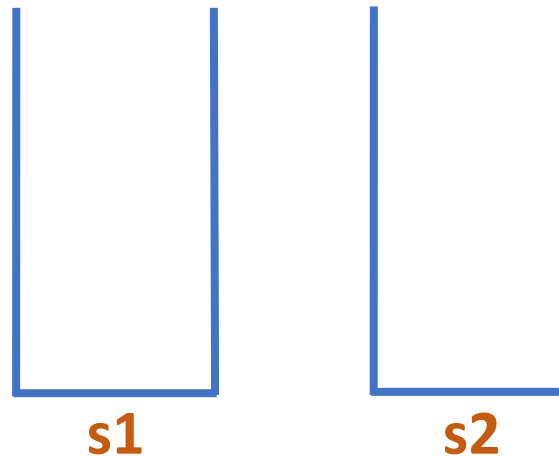
```
}
```

```
    ⋮
```

```
    current->left = 3
```



current = 5



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL)
```

```
        push(s1, current->left)
```

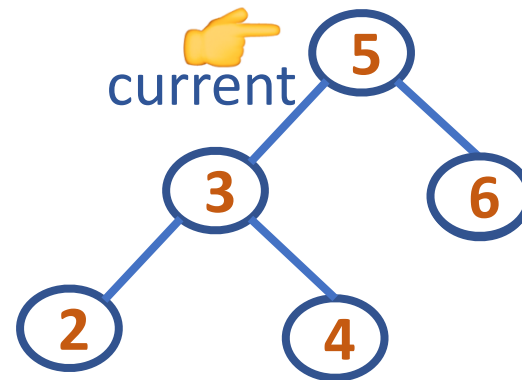
```
    if(current->right != NULL) ➡
```

```
        push(s1, current->right) ➡
```

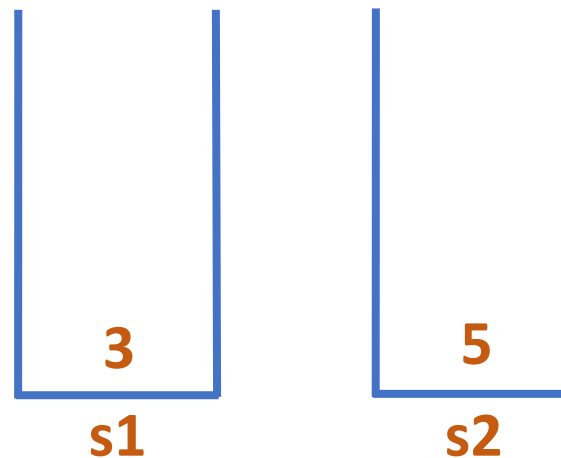
```
}
```

```
    ⋮
```

current->right = 6



current = 5



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
    while(!isEmpty(s1)) ➡
```

```
    {
```

```
        current = pop(s1) ➡
```

```
        push(s2, current)
```

```
        if(current->left != NULL)
```

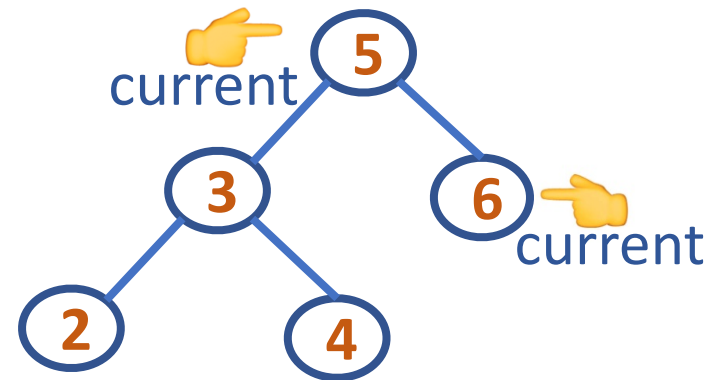
```
            push(s1, current->left)
```

```
        if(current->right != NULL)
```

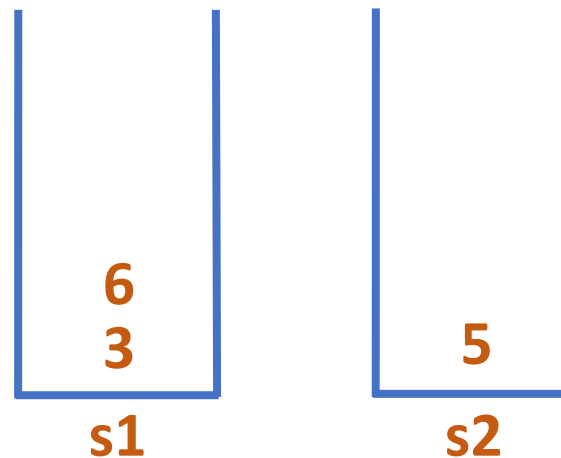
```
            push(s1, current->right)
```

```
    }
```

```
    ⋮
```



current = 5



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current) ➡
```

```
    if(current->left != NULL) ➡
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL) ➡
```

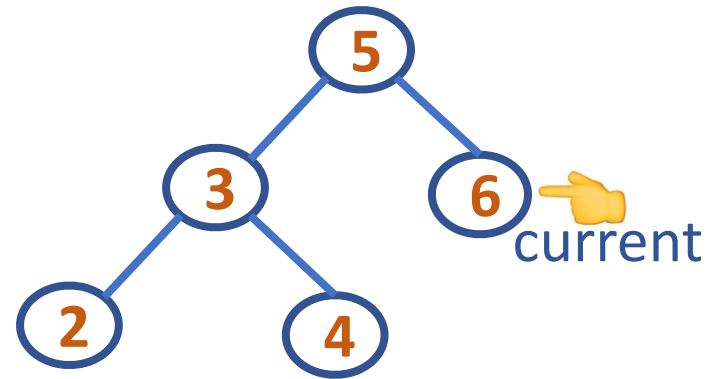
```
        push(s1, current->right)
```

```
}
```

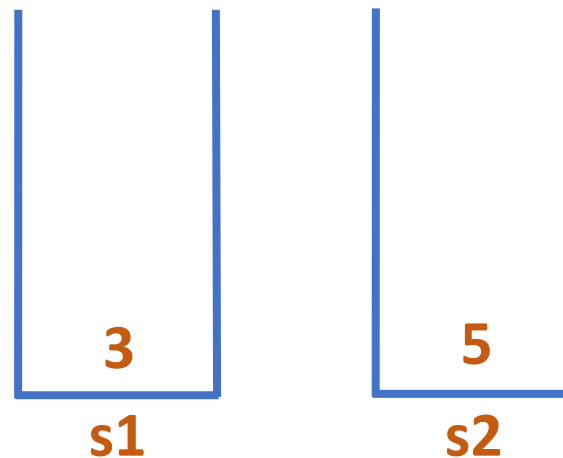
```
⋮
```

```
    current->left = N
```

```
    current->right = N
```



current = 6



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
    while(!isEmpty(s1)) ➡
```

```
    {
```

```
        current = pop(s1) ➡
```

```
        push(s2,current)
```

```
        if(current->left !=NULL)
```

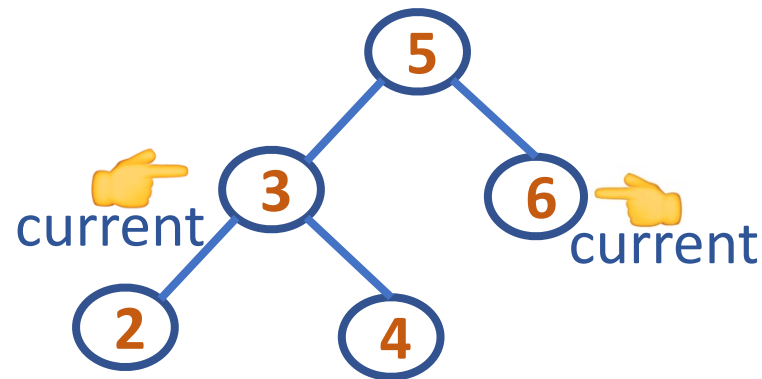
```
            push(s1, current->left)
```

```
        if(current->right !=NULL)
```

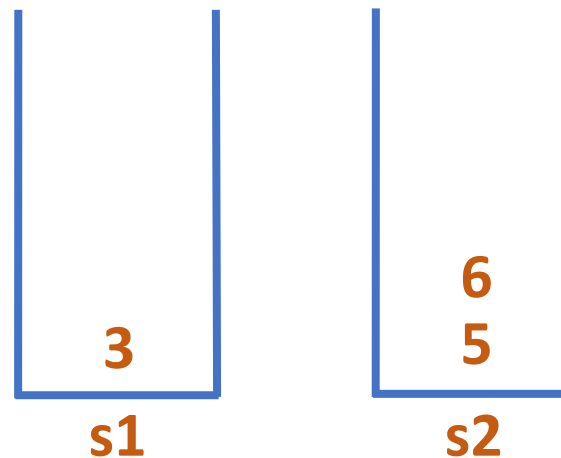
```
            push(s1, current->right)
```

```
    }
```

```
    ⋮
```



current = 6



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current) ➡
```

```
    if(current->left != NULL)
```

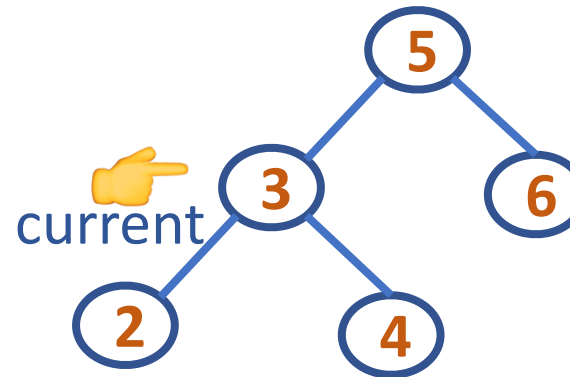
```
        push(s1, current->left)
```

```
    if(current->right != NULL)
```

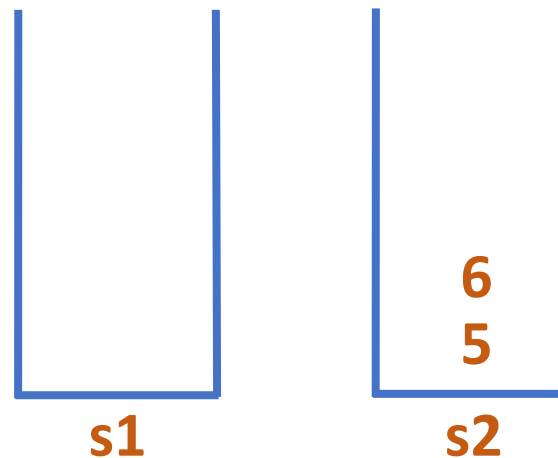
```
        push(s1, current->right)
```

```
}
```

```
    ⋮
```



current = 3



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL) ➡
```

```
        push(s1, current->left) ➡
```

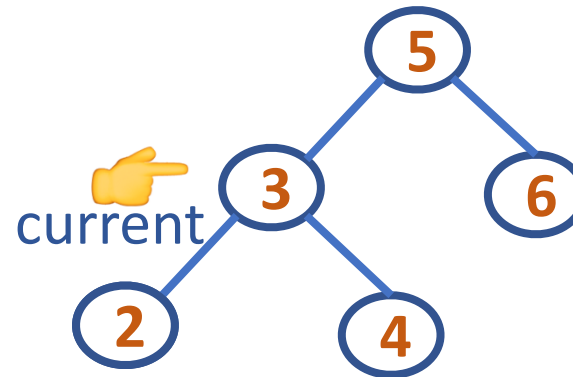
```
    if(current->right != NULL)
```

```
        push(s1, current->right)
```

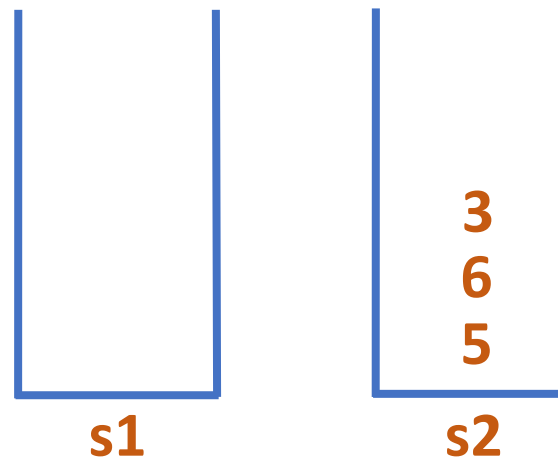
```
}
```

```
    ⋮
```

```
    current->left = 2
```



current = 3



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL)
```

```
        push(s1, current->left)
```

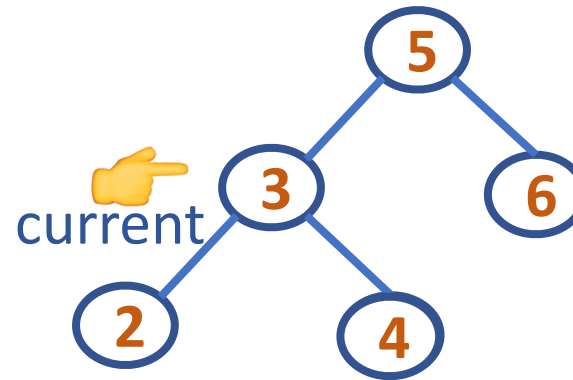
```
    if(current->right != NULL) ➡
```

```
        push(s1, current->right) ➡
```

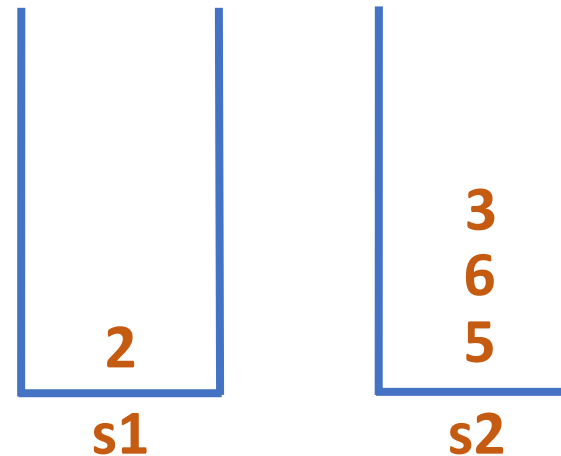
```
}
```

```
    ⋮
```

```
    current->right = 4
```



current = 3



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
    while(!isEmpty(s1)) ➡
```

```
    {
```

```
        current = pop(s1) ➡
```

```
        push(s2, current)
```

```
        if(current->left != NULL)
```

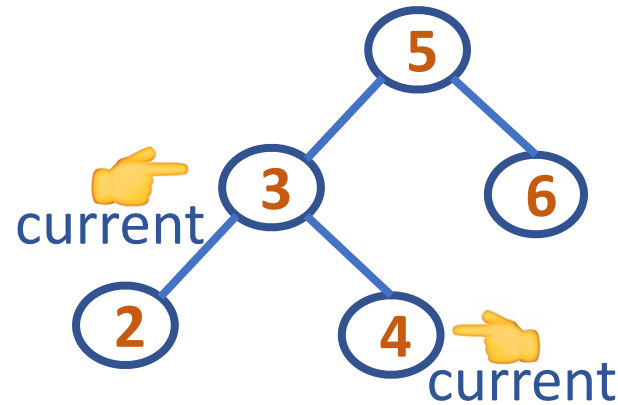
```
            push(s1, current->left)
```

```
        if(current->right != NULL)
```

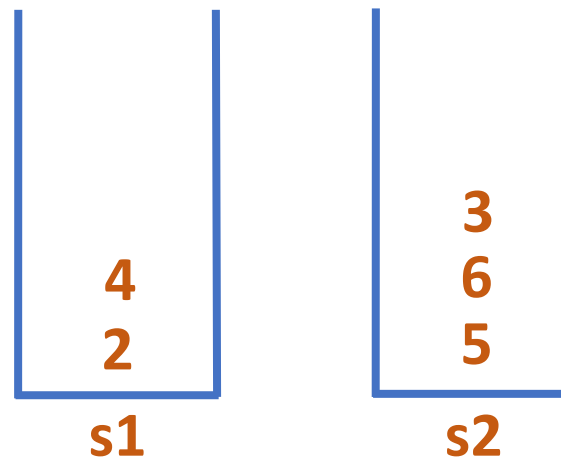
```
            push(s1, current->right)
```

```
    }
```

```
    ⋮
```



current = 3



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current) ➡
```

```
    if(current->left != NULL) ➡
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL) ➡
```

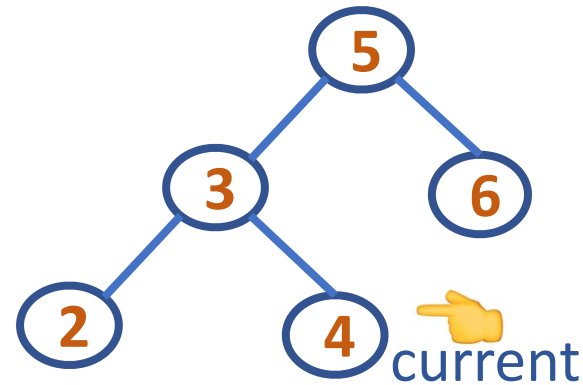
```
        push(s1, current->right)
```

```
}
```

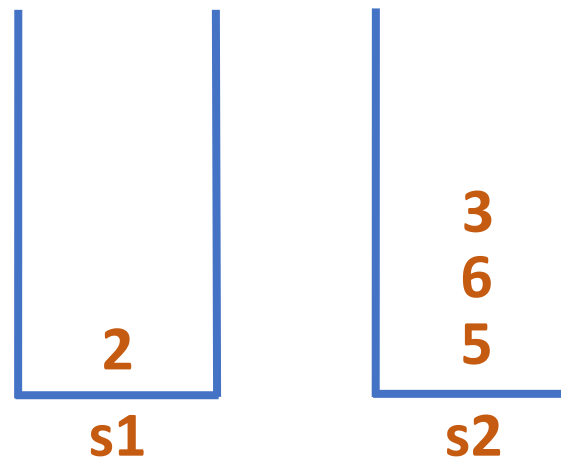
```
⋮
```

```
    current->left = N
```

```
    current->right = N
```



current = 4



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
    while(!isEmpty(s1)) ➡
```

```
    {
```

```
        current = pop(s1) ➡
```

```
        push(s2, current)
```

```
        if(current->left != NULL)
```

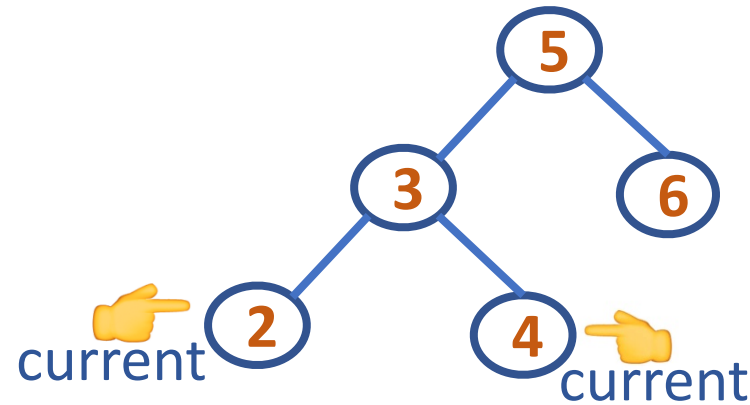
```
            push(s1, current->left)
```

```
        if(current->right != NULL)
```

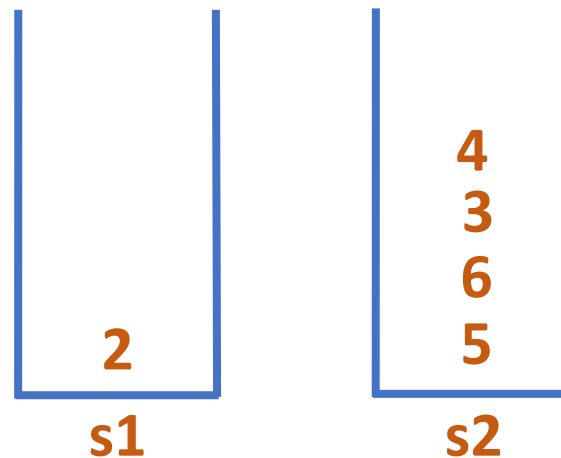
```
            push(s1, current->right)
```

```
    }
```

```
    ⋮
```



current = 4



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current) ➡
```

```
    if(current->left != NULL) ➡
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL) ➡
```

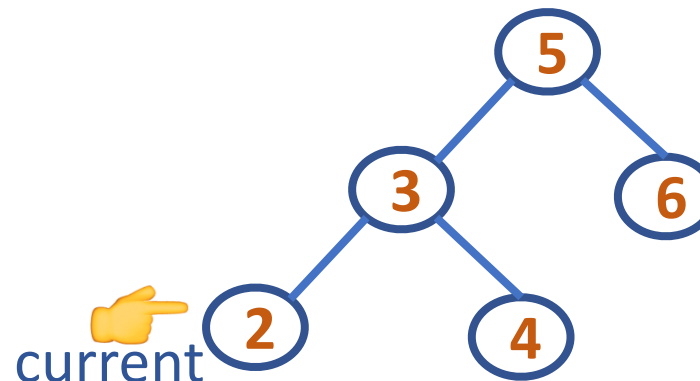
```
        push(s1, current->right)
```

```
}
```

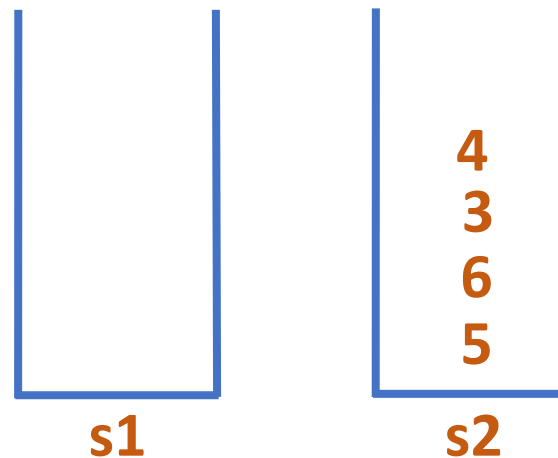
```
⋮
```

```
    current->left = N
```

```
    current->right = N
```



current = 2



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
    while(!isEmpty(s1)) ➡
```

```
    {
```

```
        current = pop(s1)
```

```
        push(s2,current)
```

```
        if(current->left !=NULL)
```

```
            push(s1, current->left)
```

```
        if(current->right !=NULL)
```

```
            push(s1, current->right)
```

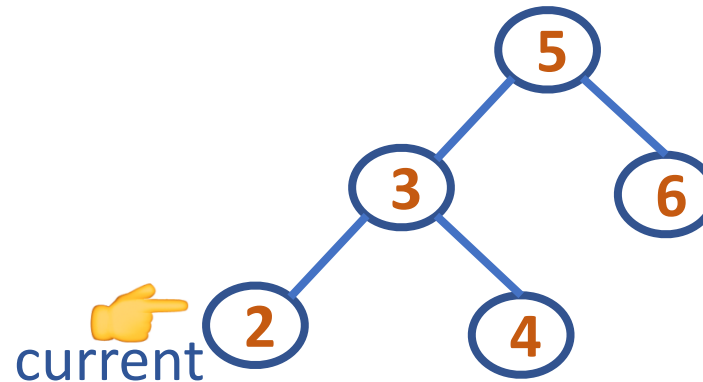
```
    }
```

```
    while(!isEmpty(s2)) { ➡
```

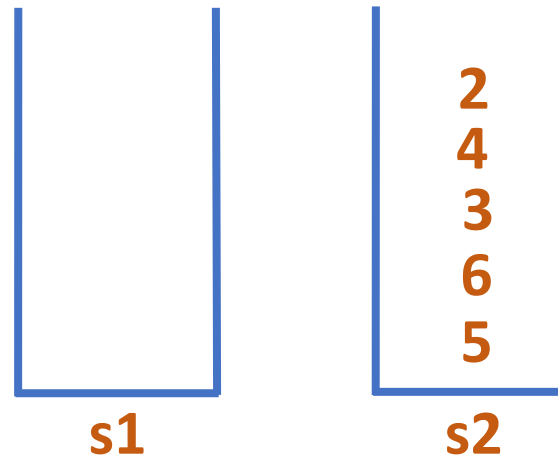
```
        current = pop(s2)
```

```
        print current->info
```

```
    }
```



current = 2



DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL)
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL)
```

```
        push(s1, current->right)
```

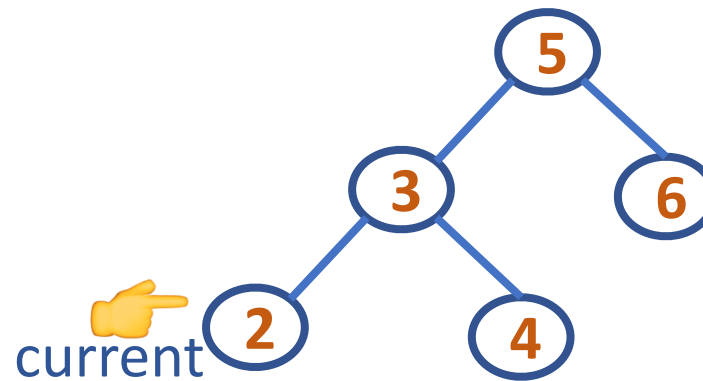
```
}
```

```
while(!isEmpty(s2)) {
```

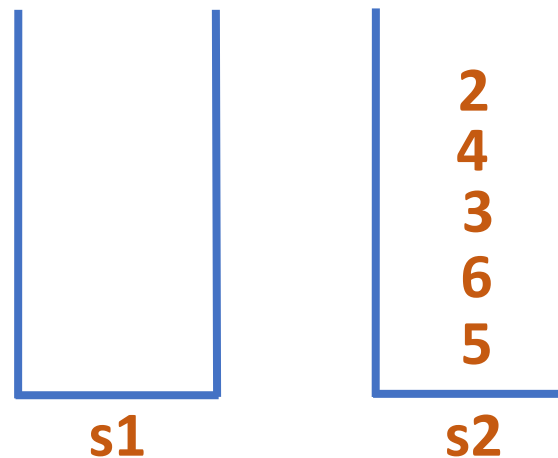
```
    current = pop(s2) ➡
```

```
    print current->info ➡
```

```
}
```



current = 2



Postorder Traversal:

2

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```

```
    ⋮
```

```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL)
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL)
```

```
        push(s1, current->right)
```

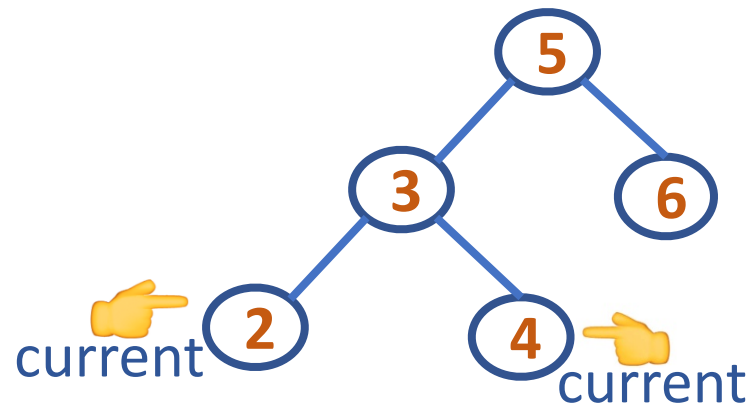
```
}
```

```
while(!isEmpty(s2)) {
```

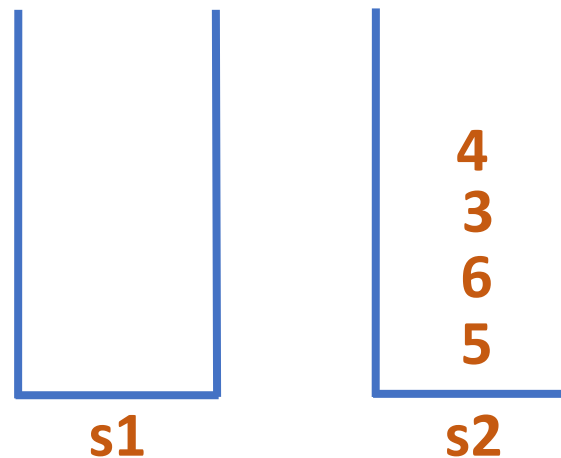
```
    current = pop(s2)
```

```
    print current->info
```

```
}
```



current = 2



Postorder Traversal:

2 4

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```



```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL)
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL)
```

```
        push(s1, current->right)
```

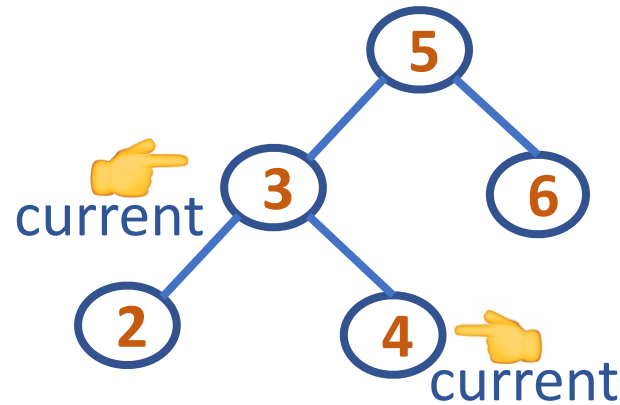
```
}
```

```
while(!isEmpty(s2)) {
```

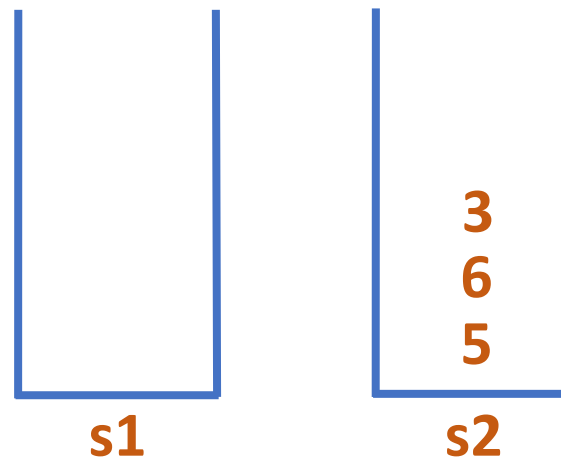
```
    current = pop(s2)
```

```
    print current->info
```

```
}
```



current = 4



Postorder Traversal:

2 4 3

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

```
iterativePostorder(root)
```



```
while(!isEmpty(s1))
```

```
{
```

```
    current = pop(s1)
```

```
    push(s2, current)
```

```
    if(current->left != NULL)
```

```
        push(s1, current->left)
```

```
    if(current->right != NULL)
```

```
        push(s1, current->right)
```

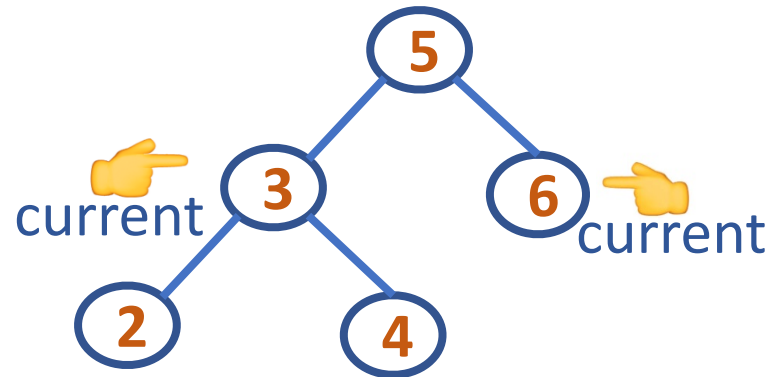
```
}
```

```
while(!isEmpty(s2)) {
```

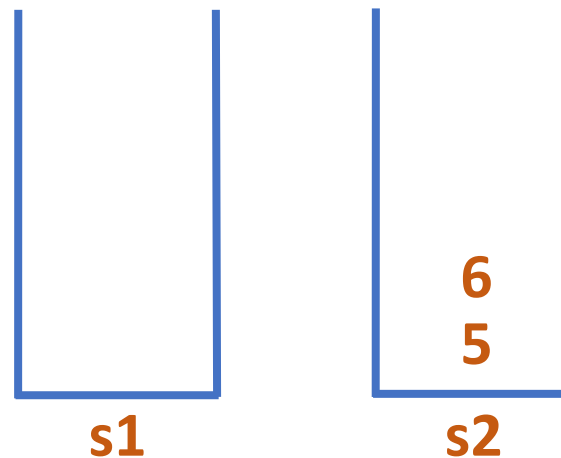
```
    current = pop(s2)
```

```
    print current->info
```

```
}
```



current = 3



Postorder Traversal:

2 4 3 6

DATA STRUCTURES AND ITS APPLICATIONS

Iterative Postorder Traversal

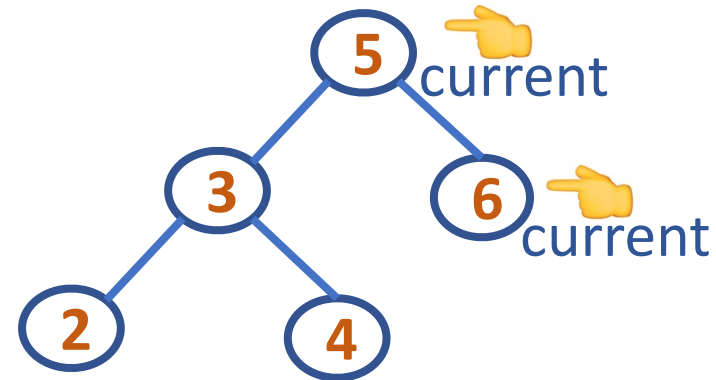
```
iterativePostorder(root)
```



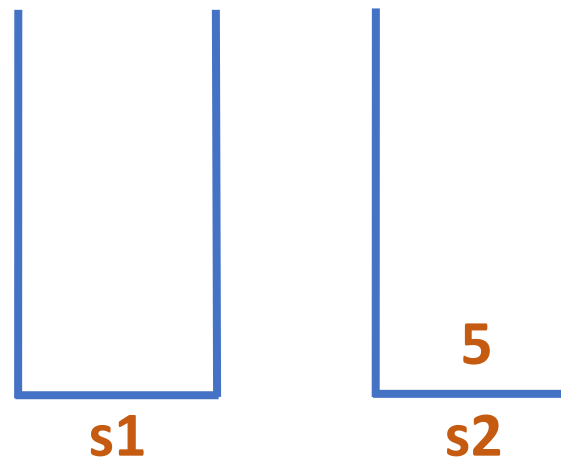
```
while(!isEmpty(s1))
```

```
{  
    current = pop(s1)  
    push(s2, current)  
    if(current->left != NULL)  
        push(s1, current->left)  
    if(current->right != NULL)  
        push(s1, current->right)  
}
```

```
while(!isEmpty(s2)) {  
    current = pop(s2)  
    print current->info  
}
```



current = 6



Postorder Traversal:

2 4 3 6 5



THANK YOU

Shylaja S S

Department of Computer Science
& Engineering

shylaja.sharath@pes.edu

+91 9449867804