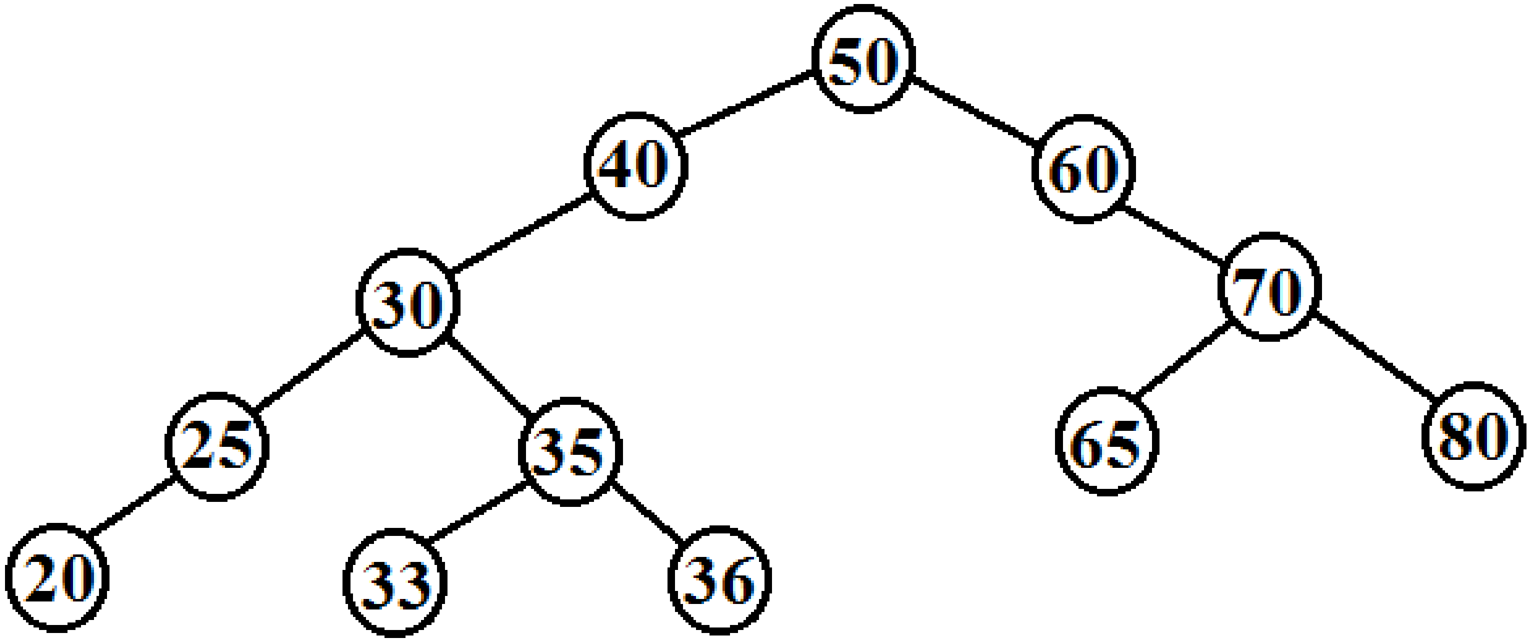
Non Recursive Traversal

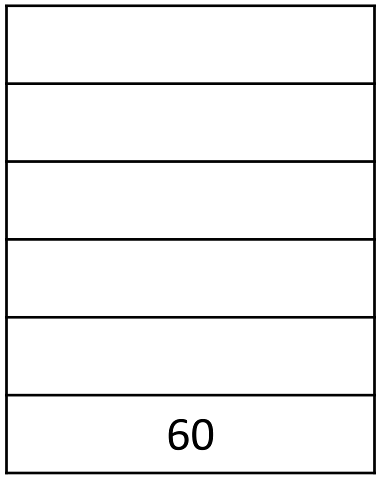
Non Recursive Preorder Traversal



```
void nrec_pre( struct node *root ){
        struct node *ptr = root;
        if( ptr == NULL){
                printf(“Tree is empty\n”);
                return;
        }
        push_stack(ptr);
        while(! stack_empty()){
                ptr = pop_stack();
                printf(“%d ”,ptr->info);
                if(ptr->rchild!=NULL){
                        push_stack(ptr->rchild);
                }
                if(ptr->lchild!=NULL){
                        push_stack(ptr->lchild);
                }
        }
        printf(“\n ”);
}
```

1. Push the root node on the stack.
2. Pop a node from the stack.
3. Visit the popped node.
4. Push right child of visited node on stack.
5. Push left child of visited node on stack.
6. Repeat step 2,3,4,5 till the stack is not empty.

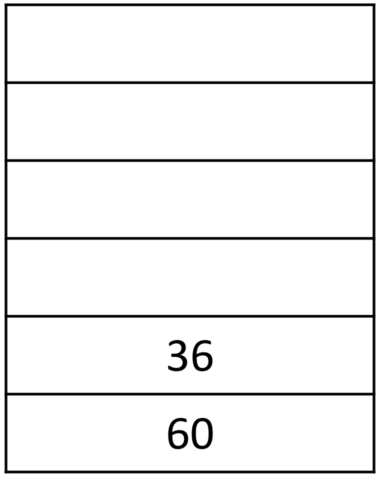
- Push 50
- Pop 50
- Push 60
- Push 40
- Pop 40
- Push 30
- Pop 30



Preorder Traversal :

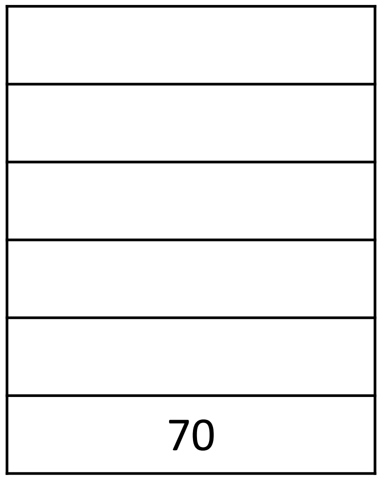
| 50 | 40 | 30 |

- Push 35
- Push 25
- Pop 25
- Push 20
- Pop 20
- Pop 35
- Push 36



Preorder Traversal :
| 50 | 40 | 30 | 25 | 20 | 35 |

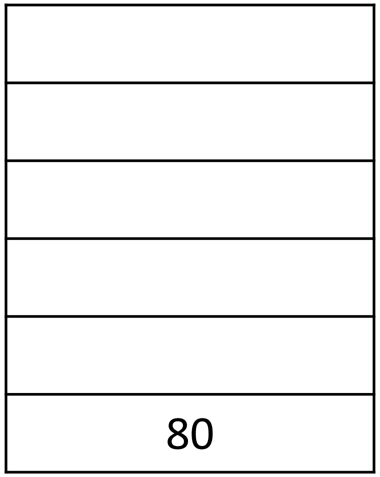
- Push 33
- Pop 33
- Pop 36
- Pop 60
- Push 70



Preorder Traversal :

| 50 | 40 | 30 | 25 | 20 | 35 | 33 | 36 |

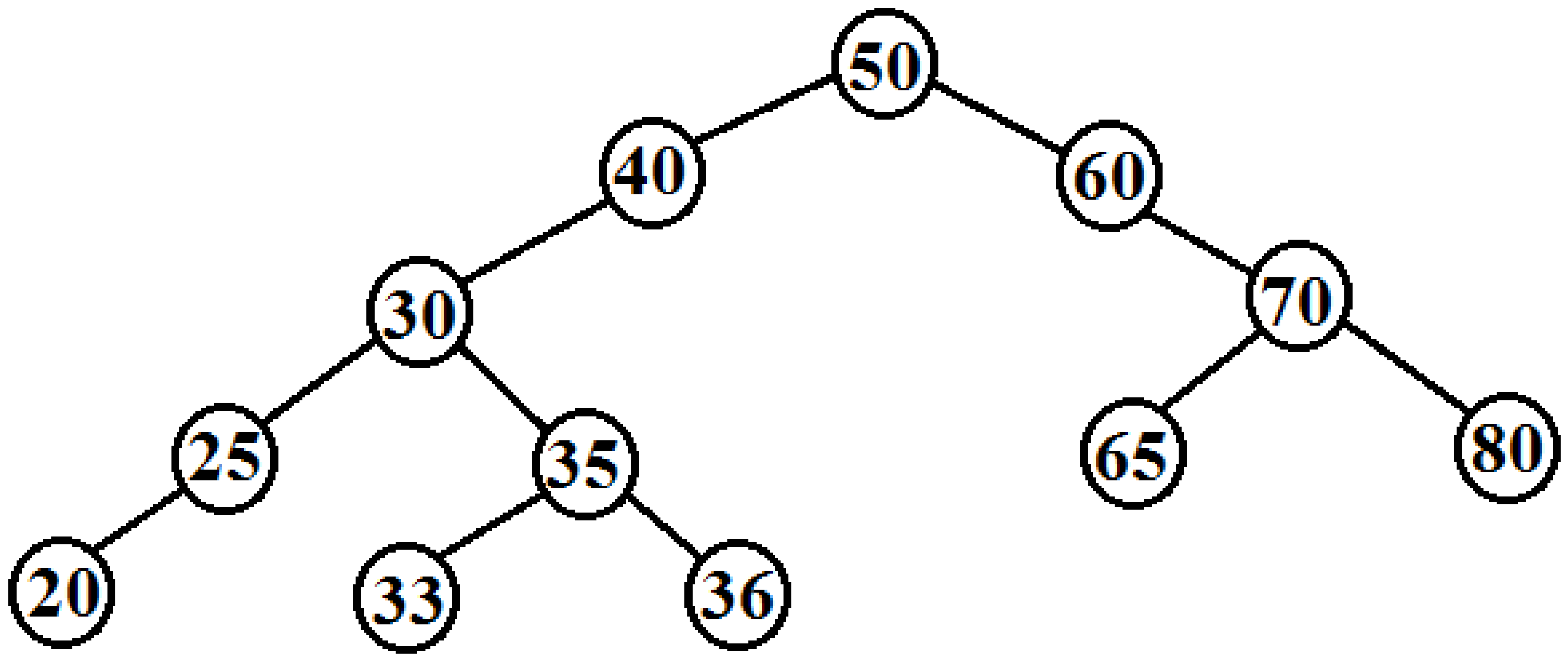
- Pop 70
- Push 80
- Push 65
- Pop 65
- Pop 80
- Stack empty



Preorder Traversal :

| 50 | 40 | 30 | 25 | 20 | 35 | 33 | 36 | 70 | 65 | 80 |

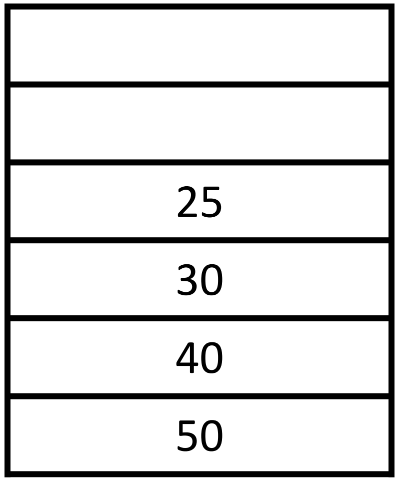
Non Recursive Inorder Traversal



```
void nrec_in( struct node *root ){
        struct node *ptr = root;
        if( ptr == NULL ){
                printf("Tree is empty\n");
                return;
        }
        while(1){
                while( ptr->lchild != NULL ){
                        push_stack(ptr);
                        ptr = ptr->lchild;
                }
                while( ptr->rchild == NULL ){
                        print("%d ",ptr->info);
                        if(stack_empty())
                                return;
                        ptr = pop_stack();
                }
                print( "%d ",ptr->info);
                 ptr = ptr->rchild;
        }
}
```

1. Initially ptr is assigned address of the root.
2. Move along leftmost path rooted at the node pointed by ptr, pushing all the nodes in the path on stack. Stop when we reach the leftmost node.
3. If node pointed by ptr has no right subtree, then visit it and pop another one from stack. Now keep on popping and visiting the node till node is popped that has a right subtree. Now ptr points to this node that has right subtree. If the stack become empty then the traversal is finished.
4. Visit the node pointed by ptr and now ptr is assigned the address of its right child

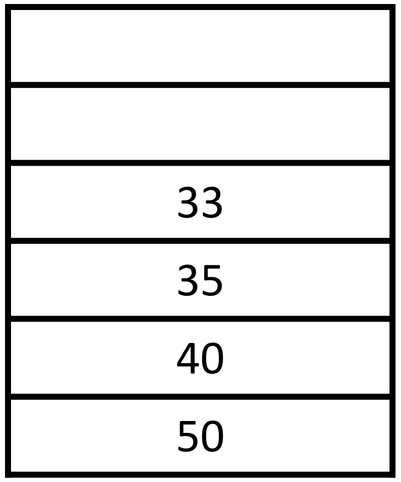
- Move along the leftmost path rooted at 50
  - Push 50
  - Push 40
  - Push 30
  - Push 25
  - Visit 20 and pop
  - Visit 25 and pop
  - Visit 30, pop and move to its right subtree



Inorder Traversal :

| 20 | 25 | 30

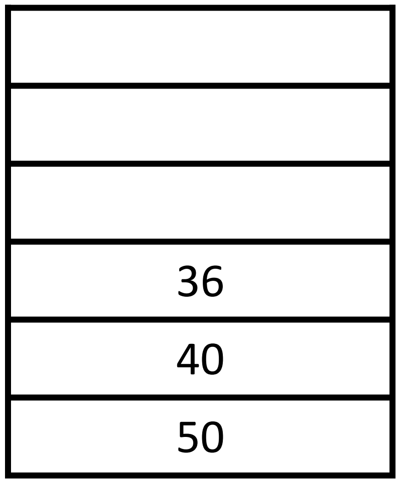
- Move along the leftmost path rooted at 35
  - Push 35
  - Visit 33 and pop
  - Visit 35 , pop and move to its right subtree



Inorder Traversal :

| 20 | 25 | 30 | 33 | 35

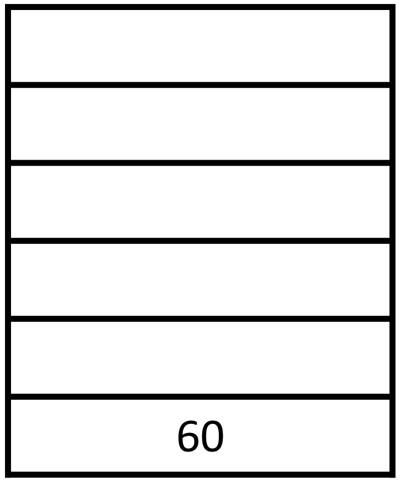
- Move along the leftmost path rooted at 36
  - Visit 36 and pop
  - Visit 40 and pop
  - Visit 50, pop and move to its right subtree



Inorder Traversal :

| 20 | 25 | 30 | 33 | 35 | 36 | 40 | 50

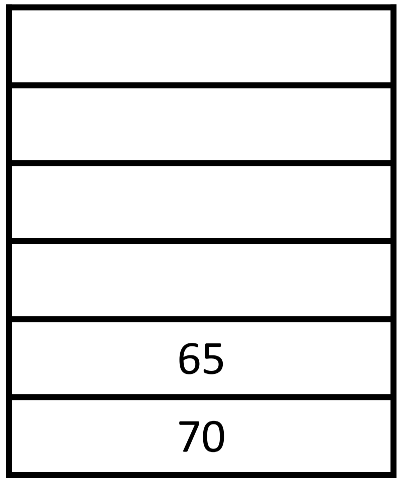
- Move along the leftmost path rooted at 60
  - Visit 60, pop and move to its right subtree



Inorder Traversal :

| 20 | 25 | 30 | 33 | 35 | 36 | 40 | 50 | 60 |

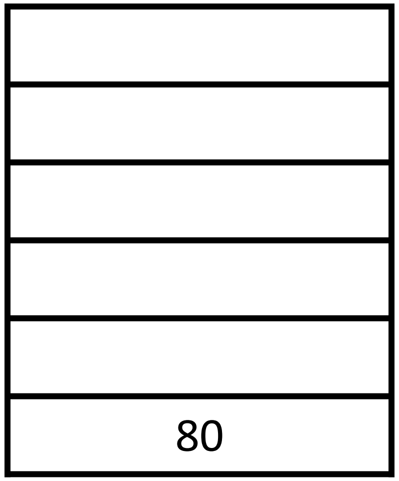
- Move along the leftmost path rooted at 70
  - Push 70
  - Visit 65 and pop
  - Visit 70, pop and move to its right subtree



Inorder Traversal :

| 20 | 25 | 30 | 33 | 35 | 36 | 40 | 50 | 60 | 65 | 70

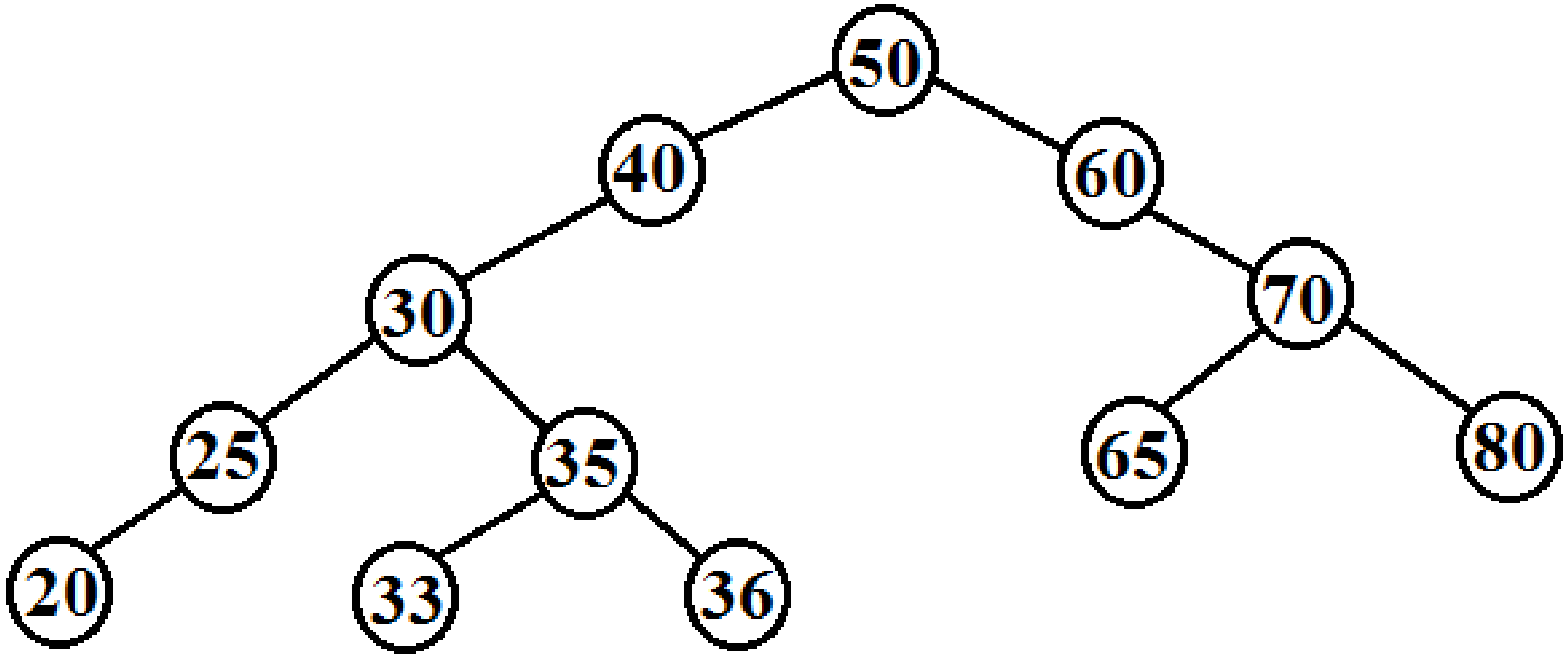
- Move along the leftmost path rooted at 80
  - Visit 80, pop and move to its right subtree



Inorder Traversal :

| 20 | 25 | 30 | 33 | 35 | 36 | 40 | 50 | 60 | 65 | 70 | 80

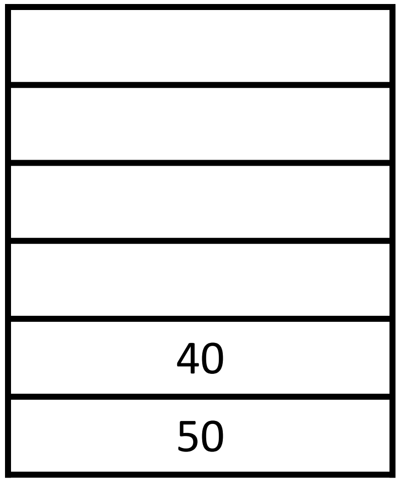
Non Recursive Postorder Traversal



```
void nrec_post( struct node *root ){
        struct node *q, *ptr = root;
        if( ptr == NULL){
                printf(“Tree is empty\n”);
                return;
        }
        q = root;
        while(1){
                while(ptr->lchild!=NULL){
                                push_stack(ptr);
                                ptr = ptr->lchild;
                }
                while( ptr->rchild == NULL | |  ptr->rchild == q ){
                                print(“%d  ”,ptr->info);
                                q = ptr;
                                if(stack_empty())
                                        return;
                                ptr = pop_stack();
                }
                push_stack(ptr);
                ptr = ptr->rchild;
        }       printf(“\n”);
}
```

1. Initially ptr is assigned address of the root node.
2. Move along leftmost path rooted at the node ptr pushing all the nodes in the path on stack. Stop when we reach the leftmost node.
3. If ptr has no right subtree or its right subtree has been traversed then visit it and pop another one from stack. Now keep on popping and visiting the node till node is popped that has a right subtree. Now ptr points to this node that has an untraversed right subtree. If the stack becomes empty then the traversal is finished.
4. Push the node pointed by ptr and now ptr is assigned the address of its right child.
5. Go to step 2.

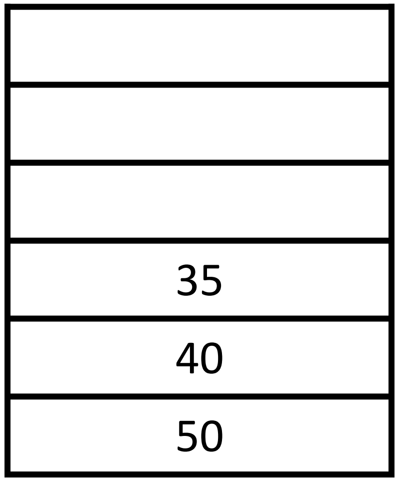
- Move along the leftmost path rooted at 50
  - Push 50
  - Push 40
  - Push 30
  - Push 25
  - Visit 20 and pop
  - Visit 25 and pop
  - Visit 30, pop and move to its right subtree



Postorder Traversal :

| 20 | 25 |

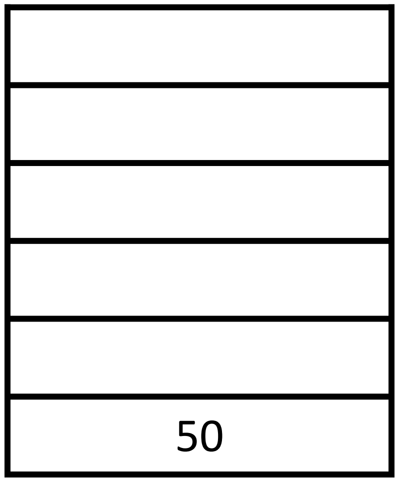
- Move along the leftmost path rooted at 35
  - Push 35
  - Visit 33 and pop



Postorder Traversal :

| 20 | 25 | 33 |

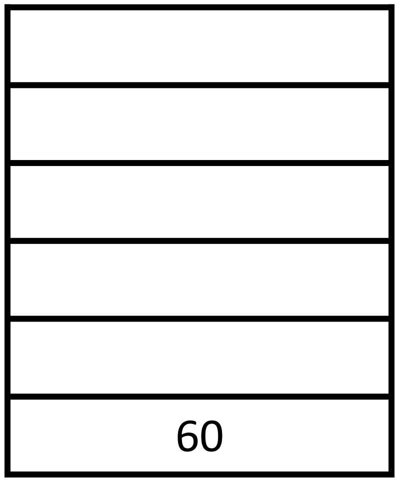
- Move along the leftmost path rooted at 36
  - Visit 36 and pop
  - Visit 35 and pop
  - Visit 30 and pop
  - Visit 40 and pop



Postorder Traversal :

| 20 | 25 | 33 | 36 | 35 | 30 | 40 |

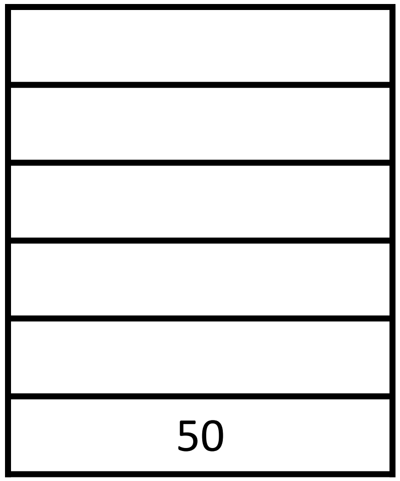
- Move along the leftmost path rooted at 60
  - Push 60
  - Push 70
  - Visit 65 and pop



Postorder Traversal :

| 20 | 25 | 33 | 36 | 35 | 30 | 40 | 65 |

- Move along the leftmost path rooted at 80
  - Visit 80 and pop
  - Visit 70 and pop
  - Visit 60 and pop
  - Visit 50 and pop
  - Stack empty



Postorder Traversal :

| 20 | 25 | 33 | 36 | 35 | 30 | 40 | 65 | 80 | 70 | 60 | 50 |

Thank You