Musab Mehadi mmehadi@jacobs-university.de

8.2) A)

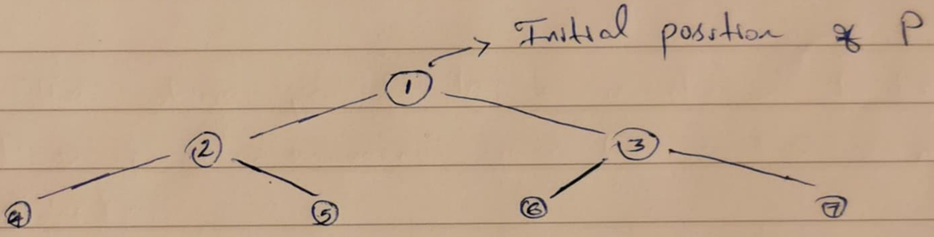
```
ReversingList (head)
{
    Node temp1 = head
    Nodes temp2 & temp3 = NULL
    while (temp1 != NULL)
    {
(saving the next node)        temp2 = temp1.next
                              temp1.next = temp3
(saving the previous node)    temp3 = temp1
                              temp = temp2
    }
    return temp3;
}
```

(A) The above pseudocode is in-situ as it doesn't involve creation of extra memory space (no creation of new arrays or nodes).

- The time complexity is $\Theta(n)$ as we can see in the while loop; it runs n times.

8.2) B)

```
                 Root          NULL
inorder (node *P , node * head)
{
    if (P = NULL)                        <- if there is no root
        return;
    static node * temp = NULL,           <- used to store previous values
    inorder (P(left), head)              <- recursively calling
    if (temp = NULL)
        head = P
    else
        P(left) = temp
        P(right) = P
        temp = P
    inorder (P(right), head)             <- recursively calling for the right side again.
```



④ Our function is iterative
- It first goes to the left most end of the tree (node 4) and goes back up sorting them on the way.