**Serie 5**

Theoretische Aufgaben

**1. Reihenfolge umkehren**

Nicht rekursiv, Zeit Θ(n), einfach verkettet, n Elemente

LIST-REVERSE(L)

nachf := NIL

zeiger := L.kopf

WHILE zeiger != nil

rechts := zeiger.nachf

zeiger.nachf := nachf

nachf := zeiger

zeiger := rechts

L.kopf = nachf

**2. Warteschlange**

**3. Schlüssel aller Knoten eines gerichteten Baums: rekursiv**

GET-KEYS(n)

IF n == NIL

RETURN

PRINT(n.key)

GET-KEYS (n.left-child)

GET-KEYS (n.right-sibling)

**4. Schlüssel aller Knoten eines gerichteten Baums: nicht rekursiv**

S = new Stack();

root = Tree.root

S.push(root)

TRAVERSE\_TREE(Node: root)

while S is not empty

current = Node

while current is not null

S.push(Node.left\_child)

current = current.left\_child

end

last\_node = S.pop()

Print(last\_node.key)

end

**5. Merge**

LIST\_MERGE(Node A, Node B)

Node first = New Node;

Node list\_end = first;

While(true)

If(A == NULL)

List\_end.next = B;

Break;

If(B == NULL)

list\_end.next = A;

break;

if(A.key <= B.key)

list\_end.next = A;

A = A.next;

else

tail.next = B;

B = B.next;

List\_end = list\_end.next;

Return first.next;