Oppgave 1

a)

Pseudokode:
Input: Et tall x.
Output: Verdien er lagt i lista.

Procedure push_back(x)

```
v <- new Node(x)
if headNode = null then
    headNode <- v
    tailNode <- v
else then
    tailNode.next <- v
    tailNode <- v
```

Input: Et tall x.
Output: Verdien er lagt i lista.

Procedure push_front(x)

```
v <- new Node(x)
if headNode = null then
    headNode <- v
    tailNode <- v
else then
    v.next <- headNode
    headNode <- v
```

Input: Et tall x.
Output: Verdien er lagt i lista.

Procedure push_middle(x)

```
v <- new Node(x)
if headNode = null then
    headNode <- v
    tailNode <- v
else then
    node <- headNode
    for i <- 0 to (size())/2 do
        node <- node.next
    v.next <- node.next
    node.next <- v
```

Input: En index i.
Output: Verdien fra den i-te indeksen av køen.

```
Procedure get(i)
    node <- headNode
    for j <- 0 to i-1 do
        node <- node.next
    print(node.data)
```

b) Javafil: Teque.java

c)

Push_back - O(1)

push_front - O(1)

push_middle - O(n)

get - O(n)

d)

Siden vi ikke har Polynomial, eksponentiell eller logaritmisk tid så har ikke begrensingene noe å si for oss.

Oppgave 2

Vanligvis er det O(log(n)) ved binærsøk, men da bruker man et ordnet array. Her bruker man en ordnet lenkeliste. A.get(i) vil gå igjennom hele lista fram til man er ved «i» hver gang man bruker metoden og det vil bli O(n). Hvis while-løkka kjører O(log(n)) og A.get(i) kjører O(n) vil dette bli O(nlog(n)).

Oppgave 3

a)

Input: En node x, der katten er plassert.
Output: Stien fra katten til roten av treet.

Procedure findPath(x)

```
v <- x
while v.parent != null do
    print(v.data)
    v <- v.parent
print(v.data)
```

b) Javafil: Kattunge.java

Oppgave 4

a)

Pseudokode:
Input : Sortert array med tall.
Output : Array med tall som er sortert i en rekkefølge som gir et balansert tre.

Procedure: balanceArray (A,x,y)

```
if x > y then
    return
int middle = (x+y)/2
print(A[middle])
balanceArray (A, middle+1, y)
balanceArray (A, x, middle-1)
```

Javafil: BalanceArray.java

b)

Pseudokode:
Input : Heap med tall.
Output : Heap med tall som er sortert i en rekkefølge som gir et balansert tre.

Procedure balanceHeap (A,x,y)

```
if x > y then
    return
int middle = (x+y)/2
leftHeap <- new heap
for int i = 0, i < middle, i++
    leftHeap.offer(heap.poll())
print(heap.poll)
BalanceHeap (heap, 0, heap.size()-1)
BalanceHeap (leftHeap, 0, leftHeap.size()-1)
```

+ Javafil: BalanceHeap.java