# Oblig 1

## Oppgave 2

### (a)Algorithm Teque:

**Input:** Et element x

**1.Procedure push\_back(x)**

backDque ← addLast(x)

if |backDque| > |frontDque|+1 then

frontDque ← addLast(backDque[0])

**2.Procedure push\_front(x)**

frontDque ← addFirst(x)

if |frontDque| > |backDque|+1 then

backDque ← addFirst(frontDque[-1])

**3.Procedure push\_middle(x)**

if (|frontDque| < |backDque|) then

frontDque ← addLast(x)

else then

backDque ← addFirst(x)

if |frontDque| > |backDque|+1 then

backDque ← addFirst(frontDque[-1])

else then

frontDque ← addLast(backDque[0])

**4.Procedure get(i)**

if (i < |frontDque|) then

return frontDque[i]

else

return backDque[i - |frontDque|]

### (c)

Verste-tilfelle til alle fire metoder er O(1),

fordi både addFirst(), addLast() og size() til en deque er O(1).

### (d)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1

## Oppgave 3

### Algorithm ReversedTre:

**Input:** En fil som beskriver et tre

**Output:**  Stien fra en viss node til rooten

kittenIndex ← fist line in the file

nodeTree ← new Node[100]

for i←0 to 100 do

nodeTree[i] ← new Node(i)

for 2nd to last line in the input file

lineArr ← line.split(" ")

foreldreIndex ← lineArr[0]

if foreldreIndex != -1 then

forelNode ← nodeTree[foreldreIndex]

for i←1 to |lineArr| do

number ← lineArr[i]

nodeTree[number].foreldre ← forelNode

peker ← nodeTree[kittenIndex]

resultat ← ""

while peker!= null do

resultat += " "+peker.data;

peker ← peker.foreldre

print resultat

## Oppgave 4

### (a)Algorithm AvlArr:

**Input:** Et sortert array med heltall

**Output:**  Et balansert søketre

Procedure printBalanced(array, start, end)

if (start > end) do

return;

mid ← (start + end) / 2;

print(arr[mid]);

printBalanced(arr, mid + 1, end);

printBalanced(arr, start, mid - 1);

arr ← empty array

for all lines in the input file do

arr ← a new array with length |arr|+1

arr[|arr|-1] = value in the line;

tree = new AvlArr();

tree.printBalanced(arr, 0, |arr|-1);

### (b)Algorithm AvlArr:

**Input:** En heap

**Output:**  Et balansert søketre

Procedure printBalanced( mainQue)

if (|mainQue| == 0) do

return;

else if (|mainQue| == 1) do

print(mainQue.poll())

else {

left ← new PriorityQueue

for i←0 to |mainQue|/2 do

left.add(mainQue.poll());

mid ← mainQue.poll();

print(mid);

printBalanced(mainQue);

printBalanced(left);