Oblig 1

Problem/Spørsmål:

Hei, jeg har brukt mye tid på oblig1 men fortatt er 2 problem som jeg ikke klarte å løse. Håper du kan se på det hvis du har tid. :)

1. Når jeg prøve kjøre cmp kommando:

java Teque < tests/oppgave2/inputs/input_100 | cmp - tests/oppgave2/outputs/output_100

Får jeg alltid feilmeling selv om retultatene matcher output.

Har prøvd legge til " " når jeg print, og har spurt gruppelæreren, men fikke ikke det til. Derfor fikk jeg ikke teste alle tests fra output_1000

2. For oblig 1-2

I oppgave2/output_100, line 7 (get 33),

forventer 208034656 (som er inserted i line 11 i input_100),

men jeg fikk 94640931 (som er inserted i line 3 i input_100).

Dette er det eneste tall som er anneledes i oppgave2/output_100;

Vet ikke hva som er feil. Har spurt gruppelæreren, og han visste ikke heller. Men jeg har laget et log til testLogO1-2.txt, som logger ut alt som kjer i min Teque. Line 89 i testLogO1-2.txt er hvor 208034656 og 94640931 møtes. Alt ser riktig ut for meg.

Kanskje noe feil etter dette??

Er det noe feil med push_middle kanskje??

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 get() er O(n), fordi size() til en deque er O(n).

Verste-tilfelle til andre tre metoder er O(1),

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

(d)

Hvis n er begrenset/en konstant, så er O(106)=O(105)=.....=O(1)

For size(), siden det er O(n), for å sørge for at det ikke blir O(1), er det viktig å ferne begrensning på N;

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);
```