INF110 Ukeoppgaver: Uke 7

OPPGAVE 1

Anta at nodene i et binært tre har en peker til forelder (null i roten), i tillegg til de vanlige venstre og hoyre. Skriv metoder lokalt i BinNode som finner (og leverer) neste node (relativt til this) i henholdsvis infiks, prefiks og postfiks rekkefølge. Null returneres dersom this er siste node i den aktuelle rekkefølgen.

OPPGAVE 2

Vi vil bruke et binært søketre til å lagre et antall objekter som inneholder en "int verdi", samt en del andre data. Vi skal anta at hver node har en venstre og en høyre peker på vanlig måte. Det kan imidlertid være flere objekter som har samme verdi (men gjerne med forskjellig data), og disse skal alle lagres som separate objekter i treet.

a) Vi kan løse dette ved alltid å gå helt til bunns når vi skal gjøre en innsetting, SELV OM vi treffer på et objekt med samme verdi på veien. Vi kan da velge alltid å gå videre ned i høyre subtre når vi passerer et objekt med samme verdi. Skriv en innsettingsmetode som følger denne filosofien, og tegn noen typiske trær denne kan gi oss.

b) I hvilken rekkefølge vil vi få ut objektene med lik verdi dersom vi skriver ut trærne produsert under a) i vanlig infiks rekkefølge? Hvordan blir dette om vi går til venstre istedenfor til høyre når vi treffer på en node med lik verdi under innsettingen?

c) Dersom det er mange noder med like verdier, kan disse bidra til at treet blir unødvendig dypt. Vi kunne da i stedet tenke på å lagre alle nodene med like verdier i en liste ut fra den første noden med denne verdien. Dette ville imidlertid kreve at vi hadde en ekstra peker for denne listen, men vi kan klare oss som følger: Når vi vil sette inn et objekt og det allerede finnes ett (men bare ett) objekt med denne verdien i treet, så hekter vi det nye objektet inn mellom det gamle, og dets høyre subtre. Dersom det senere kommer flere objekter med samme verdi, så hektes disse inn på en liste ut fra det ANDRE objektet, med venstre-pekeren som listepeker. TEGN eksempler, og skriv en innsettingsmetode som bruker denne filosofien.

d) Vis at dersom man skriver ut treet produsert under c) i innfiks rekkefølge, så kommer fremdeles nodene med like verdier samlet. Men i hvilken rekkefølge kommer de? Skriv en modifisert utskriftsmetode som gjør at noder med like verdier skrives ut i den rekkefølge de ble satt inn.

OPPGAVE 3

Vi har et ROTRETTET tre gitt ved noder av klassen:

```
class BinNode {
    BinNode opp;
    // Andre data...
}
```

der "opp" angir trestrukturen, med rot.opp = null.

Treet skal omstruktureres med en annen node i treet som rot. Omstruktureringen skal gjøres slik at noder som er forbundet med foreldre-relasjonen (opp) fortsatt skal være forbundet, men ved at forelder/barn kan være byttet.

Skriv en metode (med den nye roten som parameter) som etablerer det nye rotrettede treet.

HINT: Forsøk å beskrive nøyaktig hvilke opp-pekere som må bytte retning.

OPPGAVE 4

Vi ønsker å gå gjennom alle nodene i et binærtre ikke-rekursivt:

```
< Sett rotnoden inn på en "liste" >;
while ( < liste ikke tom > ) {
    n = < node fra listen etter gitt regel (LIFO eller FIFO)>;
  < gjør noe med n >;
  if (n.hoyre != null) {
     < Sett n.hoyre inn i listen >;
  }
  if (n.venstre != null) {
     < Sett n.venstre inn i listen >;
  }
}
```

a) Karakteriser den rekkefølgen du vil få om vi lar listen være en stakk (en LIFO-kø), og den du får om listen er en FIFO-kø

b) Hvilken rekkefølge får man om man bytter om innsettingen av n.venstre og n.hoyre over, og listen er en stakk?

c) Vis at uansett variant man velger, så vil nodene på listen alltid danne en "grenselinje" mellom de "over" (nærmere roten) som alle er ferdigbehandlet, og de "under" som ikke er behandlet. Mer presist, så vil altså følgende invariant gjelde:

   i) For hver bladnode som ER behandlet vil alle noder på veien opp mot roten også være behandlet.

ii) For hver bladnode som IKKE er behandlet, så vil nøyaktig EN av nodene på veien opp mot roten ligge på listen (dette kan være roten eller bladet selv), og alle nodene nærmere bladet er ubehandlet.