Høgskolen i Gjøvik
Avdeling for informatikk og medieteknikk

Kontinuasjonseksamen

FAGNAVN: **Algoritmiske metoder**

FAGNUMMER: **IMT2021**

EKSAMENSDATO: **8. august 2007**

KLASSE(R): **05HBINDA / 05HBINFA / 05HBISA / ……**

TID: **09.00-14.00**

FAGLÆRER: **Frode Haug**

ANTALL SIDER UTLEVERT: **4 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne.**

- Kontroller at alle oppgavearkene er til stede.
- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).
- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.
- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.
- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.
- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

Oppgave 1 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 3, 9 og 16 i læreboka.

a) Koden øverst side 28 i læreboka leser og omgjør et infix-uttrykk til et postfix-uttrykk.
Vi har infix-uttrykket: (((4 + 3) * (((2 + 3) * (3 * 2)) + 4)) * 2)
Hva blir dette skrevet på en postfix måte ?
Tegn opp innholdet på stakken etter hvert som koden side 28 leser tegn i infix-uttrykket.

b) Du skal utføre Quicksort på teksten "L A S T G A M E" (blanke regnes ikke med).
Lag en figur tilsvarende fig. 9.3 side 119 i læreboka, der du for hver rekursive sortering skriver de involverte bokstavene og markerer/uthever hva som er partisjonselementet.

c) Koden s.237 og teksten s.239 i læreboka gir oss følgende kode ved dobbel hashing:

```
const int M = 11;
int hash1(int k)   {  return  (k % M);        }
int hash2(int k)   {  return  (7 - (k % 7));  }

void insert(itemType v, infoType info)  {
   int x = hash1(v);
   int u = hash2(v);
   while (a[x].info != infoNIL)  x = (x+u) % M;
   a[x].key = v;  a[x].info = info;
}
```

"k" står for bokstavens nummer i alfabetet (1-29). Vi har en array som er 11 lang (indeks 0-10). Keyene "THELASTGAME" skal legges inn i denne arrayen. **Skriv opp hver enkelt key's returverdi fra både hash1 og hash2. Tegn også opp arrayen for hver gang en ny key legges inn.** (Innholdet i "info" trenger du ikke å ta hensyn til.)

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 30, LZW og FSM.

a) Følgende kanter i en (ikke-retted, ikke-vekted) graf er gitt:

AF CD BF ED AD FG BG EA

Vi skal nå utføre union-find m/path compression og weight balancing på denne grafen.
Tegn opp arrayen "dad"s innhold etterhvert som skogen bygges opp (jfr. fig.30.9) vha. "find"-funksjonen s.447 i læreboka. Ut fra dette: **tegn også opp den resulterende union-find skogen** (dvs. noe lignende til nedre høyre skog i fig.30.8).

b) LZW-teknikken skal brukes på følgende tekststreng: "this_is_his_thing"
Katalogen inneholder allerede alle de standard ASCII-tegnene i indeksene 0-255.
Hva blir katalogens innhold f.o.m. indeks nr.256 og oppover?
Hva blir den kodede (output) strengen?

c) Følgende deterministiske endelige tilstandsmaskin (FSM) er gitt:

$M = (\ \{q_0, q_1, q_2, q_3\},\ \{a, b\},\ \delta,\ q_0,\ q_3\ )$

Transisjonstabellen:

$\delta(q_0, a) = q_1$	$\delta(q_0, b) = q_0$
$\delta(q_1, a) = q_1$	$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_1$	$\delta(q_2, b) = q_3$
$\delta(q_3, a) = q_1$	$\delta(q_3, b) = q_0$

Tegn tilstandsmaskinen. **Hva (slags setninger/språk) godtar den ?**

Oppgave 3 (koding, 25%)

Denne oppgaven omhandler binære trær, og hvordan gjøre/sjekke om disse er *høyreskjeve*.
At de er 'høyreskjevt' vil si at *alle* nodene i treet kun har *ett* barn og dette er dets *høyre* barn.
Et slikt *meget* skjevt tre vil altså egentlig være en liste (siden ingen har venstre barn).

En node er definert som:

```
struct Node {
   int   ID;      //  Nodens ID/key/nøkkel/navn (et tall).
   Node* left;    //  Peker til venstre subtre, evt. z-noden om det er tomt.
   Node* right;   //  Peker til høyre subtre, evt. z-noden om det er tomt.
};
```

Vi har de globale variablene:

```
Node* z   = new Node;
Node* rot = z;
```

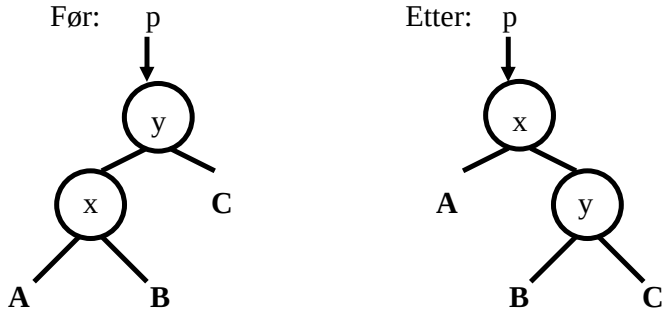
a) **Lag funksjonen** `bool er_hoyre_skjevt()`

Funksjonen sjekker om treet tilpekt av `rot` er høyreskjevt og returnerer true/false til dette.

NB: Et tomt tre er også definert som høyreskjevt.

b) **Lag funksjonen** `Node* hoyre_rotasjon(Node* p)`

Funksjonen mottar en peker til noden 'p'. Den foretar en høyrerotasjon omkring denne, slik at følgende skjer om kallet er: `p = hoyre_rotasjon(p);`



x og y er nodenes ID. A, B, C er subtrær.

c) **Lag funksjonen** `void gjor_hoyre_skjevt()`

Funksjonen omformer *hele* treet tilpekt av `rot` til å bli et fullstendig høyreskjevt binært tre.
Dette gjøres (selvsagt) vha. en mengde høyre-rotasjoner på alle nodene nedover i treet.
Du skal *ikke* bruke stakk for å løse dette, det holder med while-løkke(r)!
Hint: Bruk funksjonen du laget i oppgave 3b.

NB: *Ingen av funksjonene i oppgavene 3a, 3b og 3c skal være rekursive!*

Oppgave 4 (koding, 25%)

I en Lotto-trekning blir det blant N tall trukket ut T stk. (F.eks. i vanlig Lotto hos Norsk Tipping er N=34 og T=7, mens i Nordic Viking Lotto er N=48 og T=6.)

I slike uttrekninger forekommer det ikke sjeldent tall i numerisk sekvens rett etter hverandre.

F.eks. om rekken er (i vanlig Lotto): 7 8 12 19 25 26 27
ser vi at denne inneholder 2-er sekvensen ”7 8” og 3-er sekvensen ”25 26 27”.
Vi regner *ikke* sekvensen ”25 26 27” til også å bestå av 2-er sekvensene ”25 26” og ”26 27”.

Rekken 7 8 18 19 22 26 27 sier vi ”inneholder 2-er sekvens”. At den *egentlig* inneholder *tre* stk. er uinteressant, vi sier at den ”inneholder 2-er sekvens”.

Rekken 4 8 12 17 18 19 31 sier vi inneholder både 2-er sekvens (17 18 eller 18 19) og 3-er sekvens (17 18 19).

Din oppgave er å skrive et fullverdig program som:

- **leser inn T (*skal* være i intervallet 1-10) og N (*skal* være i intervallet T-50) fra brukeren**
- ***rekursivt* genererer *alle* mulige utvalg/rekker av T tall fra de N**
- **teller opp *totalt* antall ulike utvalg/rekker generert**
- **teller opp antall utvalg/rekker som inneholder X-er sekvens (der X er i intervallet 2 til T)**
- **til slutt skriver ut hvor mange rekker som inneholder de ulike sekvenser og deres prosentvise forekomst (av totalantallet)**

Lykke til (også med Lotto-lykken)!

frode@haugianerne.no