Høgskolen i Gjøvik
Avdeling for informatikk og medieteknikk

E K S A M E N

EMNENAVN: **Algoritmiske metoder**

EMNENUMMER: **IMT2021**

EKSAMENSDATO: **21. desember 2009**

KLASSE(R): **08HB - IND* / PUA / DRA / ISA / SPA**

TID: **09.00-14.00**

EMNEANSVARLIG: **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, 30%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap.8, 15 og 16 i læreboka.

a) Vi skal utføre Shellsort på key'ene "DRAKTENES". Jfr. kode s.109 i læreboka.
For hver gang indre for-løkke er ferdig (etter: `a[j] = v;`):
Tegn opp arrayen og skriv verdiene til 'h' (4 og 1) og 'i' underveis i sorteringen.
Marker spesielt de key'ene som har vært involvert i sorteringen (jfr. fig.8.7 s.108).

b) Legg key'ene "ARSENALDRAKTER" (i denne rekkefølge fra venstre til høyre) inn i et 2-3-4 tre. **Tegn opp treet etterhvert som bokstavene legges inn.**
Gjør også om sluttresultatet til et Red-Black tre.

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

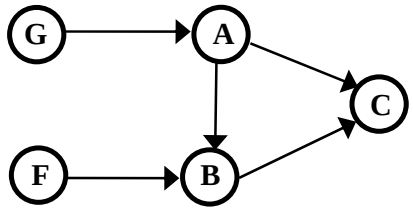
```
const int M = 17;
int hash1(int k)   {  return  (k % M);        }
int hash2(int k)   {  return  (4 - (k % 4));  }

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 17 lang (indeks 0-16). Keyene "ARSENALDRAKTER" 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, 20%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap.32, LZW og FSM.

a) Følgende rettede asykliske (ikke-vektede) graf («dag») er gitt:



Angi ***alle* mulige topologiske sorteringssekvenser** av nodene i denne grafen.

b) LZW-teknikken skal brukes på følgende tekststreng: "TIL TINE TIND TINDRE "
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:

$M = (\ \{ q_0, q_1, q_2, q_3, q_4 \},\ \{ a, b, c \},\ \delta,\ q_0,\ q_3\ )$

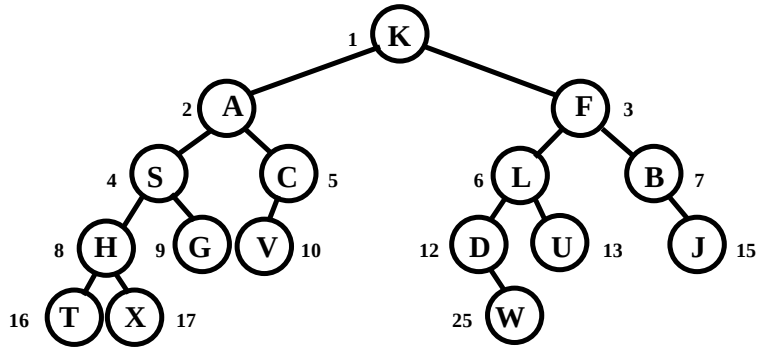
Transisjonstabellen:

$\delta(q_0, a) = q_1$	$\delta(q_0, b) = q_4$	$\delta(q_0, c) = q_4$
$\delta(q_1, a) = q_4$	$\delta(q_1, b) = q_2$	$\delta(q_1, c) = q_4$
$\delta(q_2, a) = q_4$	$\delta(q_2, b) = q_2$	$\delta(q_2, c) = q_3$
$\delta(q_3, a) = q_4$	$\delta(q_3, b) = q_4$	$\delta(q_3, c) = q_4$
$\delta(q_4, a) = q_4$	$\delta(q_4, b) = q_4$	$\delta(q_4, c) = q_4$

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

Oppgave 3 (koding, 25%)

Vi har et *binært tre* (*ikke* søketre) som ser slik ut:



Roten er alltid nummerert som nr.1. Om node nr.i har et venstre barn, så er denne nr.i*2, mens et evt. høyre barn har nr.i*2+1 – akkurat som ifm. heap i læreboka.

a) **Oppg. a1:** Når treet ovenfor traverseres på en *inorder* måte:
Skriv nodene på formen "(X, k)", der X er nodens ID og k er dets nummer.

Oppg. a2: Vi har gitt følgende om nodene i et binært tre (som beskrevet ovenfor):

X:	W	O	S	G	J	P	X	Z	B	Y	F	T	M
k:	1	2	3	4	7	14	15	28	29	56	57	114	229

Ut fra disse dataene: **Tegn det binære treet** (når de samme prinsippene som ovenfor er fulgt).

b) Nodene i vårt binære tre er representert vha:

```
struct Node  {
  char ID;          //  Nodens ID/navn (en bokstav).
  Node* left;       //  Peker til venstre subtre, evt. z-noden.
  Node* right;      //  Peker til høyre subtre, evt. z-noden.
  Node (char id, Node* l, Node* r) { ID = id; left = l; right = r; }
};
```

NB1: Legg merke til at `Node` *ikke* inneholder noen data om dets nummer (verdien 'k').

Vi har også de to globale variablene:

```
Node* z   = new Node('0', NULL, NULL);      //  z-noden.
Node* rot = z;                              //  Rot-peker.
```

Lag den *rekursive* funksjonen `void skriv(.....)`

Funksjonen skal (som i oppg. a1) sørge for at treet (som vi forutsetter er bygd opp, og at `rot` peker til dets reelle rot) traverseres på en inorder måte, og at hver node får hver sin linje med utskrift på formen "(X, k)".

NB2: Det er *ikke* lov å innføre flere data i struct'en `Node`.
Utskrift av verdien 'k' *må* derfor ivaretas vha. parameter til funksjonen.

NB3: Funksjonen startes fra `main` ved bl.a. å sende med `rot` som parameter.

c) **Lag den *rekursive* funksjonen** `bool finn(Node* p, char id)`

Funksjonen(e) skal returnere true/false til om noen node i treet har `ID` lik `id`.

NB1: Husk at treet *ikke* er et binært søketre - det er totalt usortert.

NB2: Det skal *ikke* innføres noen nye ekstra globale variable.

Oppgave 4 (koding, 25%)

En *usortert* `int`-array inneholder N telefonnumre. Disse numrene er alt mulig "rart" fra tre til åtte sifre. For at telefonering *alltid* skal være entydig, er det viktig at ingen numre er like eller prefiks av hverandre. F.eks. vil 112 være prefiks av 11246517 (*hele* det ene tallet er lik starten på det andre).

Skriv et komplett program som sjekker at alle numrene ikke er prefiks av noen andre, og som skriver ut om dette er tilfelle eller ei.

Forklar/redegjør aller først hvordan algoritmen din fungerer/er bygd opp.

Du kan forutsette at alle telefonnumrene allerede ligger i de N første "skuffene" av arrayen `tlf`.

NB: Det er *ikke* tillatt å bruke STL-kode for å løse denne oppgaven.
Men, det *er* evt. lov å henvise til kode i læreboka som kan (gjen)brukes/modifiseres litt.

Løkke tæll!

frode@haugianerne.no