Høgskolen i Gjøvik
Avdeling for informatikk og medieteknikk

E K S A M E N

EMNENAVN: **Algoritmiske metoder**

EMNENUMMER: **IMT2021**

EKSAMENSDATO: **21. desember 2010**

KLASSE(R): **09HB - 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, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap.4, 9 og 12 i læreboka.

a) **Tegn opp parse-treet** for uttrykket: ((((A + B) * (C * D)) + ((E * F) + G)) * H)
(Hint: Figur 4.4 s.41 i læreboka.)

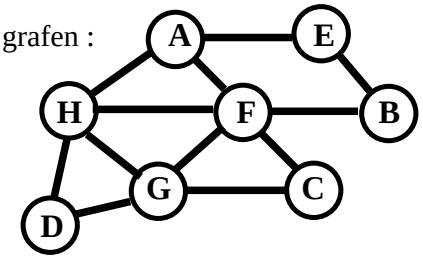
b) Du skal utføre Quicksort på teksten "H I G H G A T E" (blanke regnes ikke med).
Lag en figur tilsvarende fig. 9.3 side 119 i læreboka, der du for hver rekursive sortering skriver de involverte bokstavene og markerer/uthever hva som er partisjonselementet.

c) Vi skal utføre *rekursiv* Mergesort på key'ene "HGCEMETERY". Jfr. kode s.166 i læreboka.
For hver gang tredje og siste for-løkke i koden s.166 er ferdig:
Tegn opp arrayen med de key'ene som har vært involvert i sorteringen (jfr. fig.12.1 s.167).

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap.29, 30 og FSM.

a) Vi har grafen :



a1) **Skriv opp nabomatrisen** for dette tilfellet.

a2) **Tegn dybde-først søketreet** for dette tilfellet (dvs. ved bruk av nabomatrise), når "Visit" (s.427) fungerer som angitt i læreboka, og *vi starter i node F*.

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

FG AE EB DB DC GB HF GD

Vi skal nå utføre *union-find m/weight balancing og path compression* 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).

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

$M = (\ \{ q_0, q_1, q_2, q_3, q_4, q_5 \},\ \{ a, b \},\ \delta,\ q_0,\ \{ q_1, q_2, q_3, q_4 \}\ )$

Transisjonstabellen:

$\delta(q_0, a) = q_1$	$\delta(q_0, b) = q_3$
$\delta(q_1, a) = q_1$	$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_5$	$\delta(q_2, b) = q_2$
$\delta(q_3, a) = q_4$	$\delta(q_3, b) = q_3$
$\delta(q_4, a) = q_4$	$\delta(q_4, b) = q_5$
$\delta(q_5, a) = q_5$	$\delta(q_5, b) = q_5$

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

Oppgave 3 (koding, 33%)

Vi har et *binært søketre* bestående av nodene:

```
struct Node  {
  int   ID;       //  Nodens ID/key/nøkkel/navn (et tall).
  Node* left;     //  Peker til venstre subtre, evt. z når tomt.
  Node* right;    //  Peker til høyre subtre, evt. z når tomt.
  Node* parent;   //  Peker til oppover igjen til forelder/mor,
                                 evt. z om er rota.
  Node (int id, Node* l, Node* r, Node* p)  //  Constructor:
      { ID = id; left = l; right = r; parent = p; }
};
```

Vi har også de to globale variablene:

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

Legg merke til `parent`, som *alltid* peker til nodens mor/forelder (`rot->parent` er `z`). **I *hele* denne oppgaven skal det *ikke* innføres flere globale data eller struct-medlemmer enn det gitt ovenfor.** Ifm. de tre funksjonene du skal lage nedenfor, kan du forutsette at parameteren `n` *ikke* peker til `z`.

Hint: Tegn opp et litt større tilfeldig binært søketre, så er det lettere å studere/tenke på hvordan de ulike funksjonene skal operere.

a) **Lag den *ikke-rekursive* funksjonen** `Node* nestePreorder(Node* n)`

Funksjonen mottar pekeren `n` som parameter. Denne peker til en helt vilkårlig node ett eller annet sted i treet. Funksjonen skal returnere en peker til den *neste* noden i *preorder rekkefølge*. Er `n` selv den siste noden i treet, så returneres en peker til `z`.

Hint: Har en node venstre og/eller høyre barn, så er dette rimelig enkelt. Har den *ikke* det (altså `n` er selv en bladnode), må det letes oppover i treet igjen etter nærmeste høyre-node/-subtre som er ubesøkt.

b) **Lag den *ikke-rekursive* funksjonen** `Node* forrigePreorder(Node* n)`

Funksjonen mottar pekeren `n` som parameter. Denne peker til en helt vilkårlig node ett eller annet sted i treet. Funksjonen skal returnere en peker til den *forrige* noden i *preorder rekkefølge*. Er `n` selv rota, så returneres en peker til `z`.

Hint: Er `n` rota, et venstre barn eller mor har ingen venstre, så er dette rimelig enkelt. Er ikke noe av dette tilfelle, så er forrige i preorder rekkefølge en bladnode!

c) **Lag den *ikke-rekursive* funksjonen** `Node* nestePreorder(Node* n)`

Denne funksjonen har de samme forutsetninger, og skal gjøre det samme som funksjonen i 3a), bare at `parent` kan *ikke* brukes (pekeren er *ikke* lenger tilgjengelig).

Hint: Har en node venstre og/eller høyre barn, så er dette rimelig enkelt. Har den *ikke* det (altså er selv en bladnode), må det letes fra rota og nedover etter den *siste* noden *før* `n` som har en ubesøkt høyre-node/-subtre.

Og nok en gang: I *hele* oppgave 3 skal det *ikke* innføres flere globale data eller struct-medlemmer enn det angitt innledningsvis.

Oppgave 4 (koding, 17%)

Skriv et komplett program som skriver ut *alle* sammenhengende heltallsrekker mellom 1 og M som til sammen gir summen N. Der M er "noe mindre enn" N.

To eksempler:
N = 1000. Da vil summen av alle heltall mellom 28 og 52 være 1000, likså summen av de mellom 55 og 70, samt de mellom 198 og 202.
N = 51. Da vil følgende heltallsintervall alle gi 51: 6-11, 16-18 og 25-26.

Legg vekt på å stanse/avskjære tallutregningen når M har blitt så stor at videre summering av minst to tall uansett vil gi totalsum større enn N.

Løkke tæll!

frode@haugianerne.no