Høgskolen i Gjøvik
Institutt for informatikk og medieteknikk

E K S A M E N

FAGNAVN: **Algoritmiske metoder**

FAGNUMMER: **IMT2021**

EKSAMENSDATO: **4. desember 2006**

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: (((6 + 4) * (3 + 2)) * (((3 + 3) + (1 + 3)) * 5))
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 "E M I R A T E S" (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:

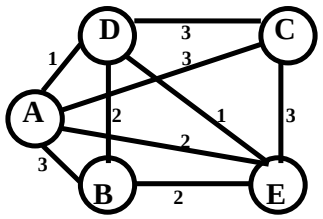
```
const int M = 17;
int hash1(int k)   {  return  (k % M);        }
int hash2(int k)   {  return  (5 - (k % 5));  }

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 "EMIRATESSTADIUM" 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. 31, LZW og FSM.

a) Følgende vektede (ikke-rettede) graf er gitt:



Hver nodes naboer er representert i en naboliste. Alle listene er *sortert alfabetisk*.
Tegn opp minimums spenntreet for denne grafen, etter at koden s.455 i læreboka er utført (der "priority" er lik "t->w").

Tegn også opp innholdet i prioritetskøen etterhvert som konstruksjonen av minimums spenntreet pågår (jfr. fig.31.4 i læreboka). **NB:** Husk at ved lik prioritet så vil noden sist oppdatert (nyinnlagt eller endret) havne først i køen.

b) LZW-teknikken skal brukes på følgende tekststreng: ”ManU_ManU_UUU_UUU”
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:

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

Transisjonstabellen:

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

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

Oppgave 3 (koding, 25%)

Denne oppgaven omhandler binært søketre, og hvordan finne avstanden (dvs. lengden på stien / antall kanter) mellom to noder.

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;
```

Vi har også gitt (koden for) funksjonen:

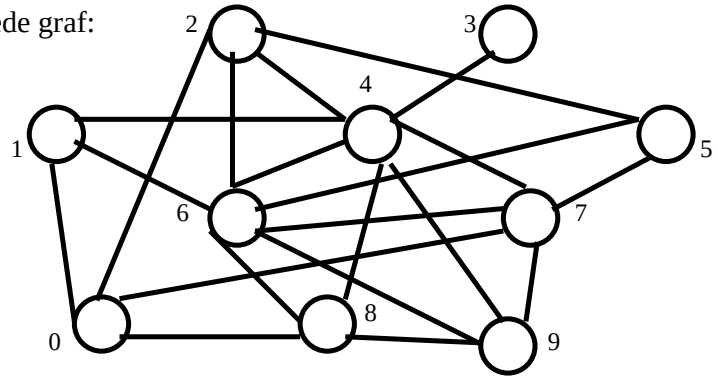
```
int hoyde(Node* p, int id)  {       //  Returnerer (om mulig) høyden fra
  int hoy = 0;                      //    node 'p' og ned til verdien 'id',
  while (p != z)  {                 //    Om 'id' ikke finnes returneres -1.
    if (p->ID == id)  return hoy;
    p = (id < p->ID) ? p->left : p->right;
      hoy++;
  }
  return -1;
}
```

Lag funksjonen `int avstand(int id1, int id2)`
Funksjonen returnerer (om mulig) avstanden mellom nodene med 'id1' og 'id2'.
Om minst en av verdiene *ikke* finnes i det binære søketreet, så returneres -1.

NB: - Funksjonen skal *ikke* være rekursiv.
- Bruk funksjonen `hoyde(…)` aktivt.
- Alle nodene i søketreet er unike (har ulik ID).
- Om 'id1' og 'id2' er like, så returneres (selvsagt) verdien 0.
  Men ikke spesialbehandle dette tilfellet!

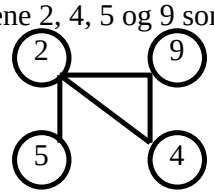
Oppgave 4 (koding, 25%)

Vi har følgende urettede/uvektede graf:



Tallene ved nodene er *ikke* deres id/navn/nummer, men bare en referanse til dem.
Utfordringen er å plassere (også) tallene/sifrene 0-9 som id/navn/nummer "*inni*" hver node.
Kravet til denne "navnsettingen" er at følgende blir oppfylt:
0: 21 1:33 2:14 3:27 4: 1 5:14 6:20 7:12 8:28 9:20
Dvs. tar man summen av id'ene (numrene) for naboene til den med id nr.0 (det er *ikke* den med *referanse* nr.0 ovenfor) får man 21, tar man summen av id'ene for naboene til den med id nr.1 får man 33, osv.

Eks (om det *kun* er tallene 2, 4, 5 og 9 som skal plasseres som id i en mindre/annen graf):



Så vil følgende være oppfylt: 2: 18 (4+5+9) 4: 11 (2+9) 5: 2 9: 6 (2+4)

Oppgave går ut på å skrive et komplett program (med hjelpearray(er) og funksjoner) som finner/skriver ut "alle" svarene på denne problemstillingen. *Om* du oppdager at du kan bruke kode *direkte* fra læreboka, eksemplene (EKS_xx.CPP) eller andre løsningsforslag i faget, så er det bare å *henvise* til dette (altså: *ikke* skrive av). Programmet bør nok bl.a. være basert på en rekursiv løsning!
Du kan forutsette at grafens nabomatrise er ferdig definert i arrayen `int adj[10][10]`.

Lykke til!

frode@haugianerne.no