Eksamen INF-1100
Innføring i programmering og datamaskiners virkemåte
Vår 2012

Eksamenssettet består av 4 oppgaver.

Der oppgaven ber om at du skriver en funksjon kan du bruke C lignende pseudokode. Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

Oppgave 1 - 20%

(a) Oversett følgende tall fra desimal til binær representasjon:

- 34
- 17

```
Løsningsforslag 1a:

34 = 100010
17 = 10001
```

(b) Adder sammen følgende binære tall og oversett resultatet til desimal representasjon.

- 00001010 + 00100101
- 00101010 + 00000011

```
Løsningsforslag 1b:
00001010 + 00100101 = 47
00101010 + 00000011 = 45
```

(c) Skisser i pseudo-kode hvordan du oversetter et tall fra desimal til binær representasjon.

Løsningsforslag 1c:

```
void desimaltilbinær(int tall)
{
    int bit;
    unsigned int mask;

    mask = 1 << 31;  Alternativt: mask = 1 << (sizeof(int)*8 - 1)
    while (mask != 0) {
        bit = 0;
        if (tall & mask)
            bit = 1;
        lagre eller skriv ut 'bit'
        mask = mask >> 1;
    }
}
```

(d) Skisser i pseudo-kode hvordan du oversetter et tall fra binær til desimal representasjon.

```
Løsningsforslag 1c:

Antar 32 bit binærtall og 2'er komplement representasjon

int tall;
int ernegativt;
int mask;

ernegativt = 0;
if binærtall_31 == 1:
    ernegativt = 1;
    binærttall = NOT(binærtall) + 1;

tall = 0;
mask = 1;
for i = 0..31:
    if binærtall_i & mask == 1:
        tall = tall + mask;
    mask = mask << 1;

if ernegativt == 1:
    tall = NOT(tall) + 1;
```

Oppgave 2 - 25%

De fleste av dagens datamaskiner er strukturert i henhold til en modell foreslått av John von Neumann i 1946. Beskriv komponentene i denne modellen og kommunikasjonen mellom disse (maksimum 2 sider).

Oppgave 3 - 30%

Gitt et array A som inneholder n heltall.

(a) Skisser i pseudo-kode en funksjon som finner tallet i A med størst verdi.

```
Løsningsforslag 3a:

int størst(int *A, int len)
{
    int maks;

    maks = 0;
    for (i = 1; i < len; i++) {
        if (A[i] > A[maks])
            maks = i;
    }
    return A[maks];
}
```

(b) Skisser i pseudo-kode en funksjon som beregner *medianen* av tallene i A. En algoritme du kan bruke er å først sortere alle tallene i A. Deretter kan medianen beregnes som følger: dersom n er er ett oddetall er medianen det midterste elementet. Dersom n er et partall, er medianen gjennomsnittet av de to midterste tallene. Du kan anta at det eksisterer en funksjon for å sortere tallene i A.

Løsningsforslag 3b:

```
int median(int *A, int len)
{
    int mid;

    sorter(A, len);

    mid = len/2;
    if ((len %2) == 1)
        return A[mid];
    else
        return (A[mid-1] + A[mid])/2;
}
```

(c) Anta at tallene i A representerer høyden til personer i en bestemt aldersgruppe. Skisser i pseudo-kode en funksjon som finner et intervall som omfavner 95% av tallene i A (95% av alle tallene i A må ha en verdi som er innenfor dette intervallet). Hint: sorter tallene i A først.

Løsningsforslag 3c:

```
void intervall(int *A, int len)
{
    int minidx, maxidx;

    sorter(A, len);

    minidx = (len*2.5)/100;
    maxidx = (len*97.5)/100;

    // Intervallet er gitt ved tallene i A[minidx] og A[maxidx]
}
```

Oppgave 4 - 25%

Begge oppgavene nedenfor involverer lister med et ukjent antall elementer. Elementene har en assosiert nøkkel og du kan anta at det eksisterer en funksjon *sammenlign* som tar to elementer som argument, sammenligner elementenes nøkler og returnerer en verdi som indikerer forholdet mellom nøklene:

```
int sammenlign(void *e1, void *e2)
```

sammenlign vil returnere -1 dersom nøkkel til $e1$ har mindre verdi enn nøkkel til $e2$, 0 dersom nøkkel til $e1$ og nøkkel til $e2$ har lik verdi og 1 dersom nøkkel til $e1$ har større verdi enn nøkkel til $e2$.

(a) Gitt en liste a. Skisser i pseudo-kode en funksjon som finner ut om det er to elementer i a med nøkler som har lik verdi.

Løsningsforslag 4a:

```
int likverdi(list_t *a)
{
    list_iterator_t *iter, *iter2;
    void *item, *item2;

    iter = list_createiterator(a);
    item = list_next(iter);
    while (item != NULL) {
        iter2 = list_createiterator(a);
        item2 = list_next(iter2);
        while (item2 != NULL) {
            if (item != item2 && sammenlign(item, item2) == 0)
                 break;
            item2 = list_next(iter2);
        }
        list_destroyiterator(iter2);

        // Like verdier funnet?
        if (item2 != NULL)
            break;

        item = list_next(iter);
    }
    list_destroyiterator(iter);

    // Like elementer funnet dersom item != NULL
    if (item != NULL)
        return 1;
```

```
        else
            return 0;
    }
```

(b) Gitt to lister a og b. I begge listene har et element i posisjon i en nøkkel med mindre eller lik verdi sammenlignet med elementet i posisjon $i + 1$. Elementene i listene er med andre ord i sortert rekkefølge (lav til høy). Skisser i pseudo-kode en funksjon som tar a og b som argument og returnerer en ny liste med elementene fra a og b i sortert rekkefølge (lav til høy).

Løsningsforslag 4b:

```
list_t *slåsammen(list_t *a, list_t *b)
{
    list_t *new;
    list_iterator_t *iter_a, *iter_b;
    void *item_a, *item_b;

    new = list_create();

    iter_a = list_createiterator(a);
    iter_b = list_createiterator(b);

    item_a = list_next(iter_a);
    item_b = list_next(iter_b);
    while (item_a != NULL || item_b != NULL) {
        if (item_a == NULL) {
            // a liste er tom. Flytt fra b
            list_addfirst(new, item_b);
            item_b = list_next(iter_b);
        } else if (item_b == NULL) {
            // b liste er tom. Flytt fra a
            list_addfirst(new, item_a);
            item_a = list_next(iter_a);
        } else {
            // Både a og b har elementer igjen
            if (sammenlign(item_a, item_b) < 0) {
                // a nøkkel er mindre enn b. Flytt fra a
                list_addfirst(new, item_a);
                item_a = list_next(iter_a);
            } else {
                // a og b har like nøkler, eller b nøkkel er mindre enn a.
                // Flytt fra b
                list_addfirst(new, item_b);
                item_b = list_next(iter_b);
            }
        }
    }
    return new;
```

```
    }
```

Du kan anta at følgende listefunksjoner er tilgjengelige:

```
// Lag en ny liste
list_t *list_create(void);

// Fjern og returner første element i en liste
void *list_removefirst(list_t *list);

// Sett inn et element først i en liste
int list_addfirst(list_t *list, void *item);

// Lag en ny listeiterator som peker på første element i listen
list_iterator_t *list_createiterator(list_t *list);

// Returner element som pekes på av iterator og
// la iterator peke på neste element.  NULL
// returneres når en når slutten på listen.
void *list_next(list_iterator_t *iter);

// Frigi iterator
void list_destroyiterator(list_iterator_t *iter);
```