# Eksamen INF-1100
# Innføring i programmering og datamaskiners virkemåte
# Høst 2011

***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%

Anta at symbolet ^ representerer den logiske operasjonen *eksklusiv or*. Følgende tabell viser resultatet av ^ operasjonen mellom to binære siffer:

```
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0
```

Gitt to variabler $x$ og $y$. Anta at $x$ har verdien $3$ og $y$ verdien $5$. Hvilke verdier har $x$ og $y$ **etter** at følgende sekvens av operasjoner er utført?

```
x = x ^ y
y = x ^ y
x = x ^ y
```

```
Løsningsforslag 1:

Det holder å si at y vil ha verdien 3 og x vil ha verdien 5, siden
oppgaven ikke ber om utregning.  Denne sekvensen kan generelt benyttes
for å bytte to variabler uten å bruke en tredje variabel som mellomlager.
```

## Oppgave 2 - 25%

De fleste av dagens datamaskiner er strukturert i henhold til en modell foreslått av John von Neumann i 1946.

**a)** Gi en kort beskrivelse av komponentene i von Neumann modellen.

**b)** Beskriv kort hvordan I/O utføres i en von Neumann-basert datamaskin.

# Oppgave 3 - 30%

Gitt et array $A$ som inneholder $n$ heltall.

**(a)** Skisser i pseudo-kode en funksjon som organiserer tallene i $A$ slik at oddetall kommer før partall.

Løsningsforslag 3a:

```
void organiserer(int *A, int lengde)
{
    int x, y;
    int tmp;

    for(x = 0; x < lengde; x++) {
        // Finn partall
        if(A[x] % 2 == 0) {
            // Let etter oddetall og bytt med partall
            for(y = x; y < lengde; y++) {
                if(A[y] % 2 == 1) {
                    tmp = A[y];
                    A[y] = A[x];
                    A[x] = tmp;
                }
            }
        }
    }
}
```

**(b)** Skisser i pseudo-kode en funksjon som sorterer tallene i $A$ i stigende rekkefølge.

Løsningsforslag 3b:

```
void sorterer(int *A, int lengde)
{
    int x, y, tmp;

    for(x = 0; x < lengde; x++) {
        for(y = x; y < lengde; y++) {
            if(A[y] < A[x]) {
                tmp = A[y];
                A[y] = A[x];
                A[x] = tmp;
            }
        }
    }
}
```

**(c)** $A$ har et *majoritetselement* dersom et tall forekommer mer enn $n/2$ ganger i $A$.

Følgende algoritme kan benyttes for å avgjøre om $A$ har et majoritetselement. Anta at vi først sorterer $A$. Dersom $A$ har et majoritetselement må dette tallet være i $A[n/2]$. Dersom antall forekomster av $A[n/2]$ i $A$ er høyere enn $n/2$, er $A[n/2]$ et majoritetselement.

Skisser i pseudo-kode en funksjon som avgjør om A har et majoritetselement. Her kan du gjenbruke sorteringsfunksjonen fra b).

Løsningsforslag 3c:

```
int harMajoritetselement(int *A, int lengde)
{
    int x;
    int tall;
    int count;

    // Sorterer tallene.  Dersom det finnes et majoritetselement må dette ligge
    // i midten av arrayet etter sortering
    sorterer(A, lengde);

    // Tell opp antall forekomster av tallet i midten av arrayet
    count = 0;
    tall = A[lengde/2];
    for(x = 0; x < lengde; x++)
        if(A[x] == tall)
            count++;

    if(count > lengde / 2)
        return 1;
    return 0;
}
```

# Oppgave 4 - 25%

**(a)** Skisser i pseudo-kode en funksjon som tar en liste som argument og returnerer en ny liste med de samme elementene som i argument-listen, men i reversert rekkefølge.

Løsningsforslag 4a:

```
list_t *nyReversListe(list_t *list)
{
    list_t *nyListe;
    list_iterator_t *iter;
    void *item;

    nyListe = list_create();
    iter =  = list_createiterator(list);
    item = list_next(iter);
    while (item != NULL) {
        list_addfirst(nyListe, item);
        item = list_next(iter);
    }

    return nyListe;
}
```

**(b)** Skisser i pseudo-kode en funksjon som finner det $k$ **siste** element i en liste. For eksempel, dersom listen har $10$ elementer og $k$ er lik $3$, skal element nummer $7$ returneres. Du kan anta at $k$ har en verdi mindre eller lik lengden på listen.

Løsningsforslag 4b:

```
void *getKLastElement(list_t *list, int k)
{
    list_iterator_t *iter;
    void *item;
    int count;
    int i;


    // Tell antall elementer i listen
    iter = list_createiterator(list);
    item = list_next(iter);
    count = 0;
    while (item != NULL) {
        count++;
        item = list_next(iter);
    }
    list_destroyiterator(iter);

    // Reset iterator og tell frem til k siste element
    iter = list_createiterator(list);
    for(i = 0; i < count-k; i++)
        item = list_next(iter);
    list_destroyiterator(iter);
    return item;
}
```

Du kan anta at følgende listefunksjoner er tilgjengelige:

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

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

// 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);
```