Eksamen INF-1100
Innføring i programmering og datamaskiners virkemåte
Vår 2013

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%

Hvilken verdi vil funksjonen *ukjent* nedenfor returnere ved følgende kall:

a) *ukjent(16)*

```
Løsningsforslag 1a:
Funksjonen vil returnere 4
```

b) *ukjent(3)*

```
Løsningsforslag 1b:
Funksjonen vil returnere 1
```

```
int ukjent(int a)
{
    int b;

    b = 4;
    while ((a & (1 << b)) == 0) {
        b = b - 1;
    }
    return b;
}
```

Oppgave 2 - 25%

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 - 20%

Følgende program er ment å skrive ut '-' 20 ganger, men det er en feil en plass..

```
#include <stdio.h>
int main()
{
    int i;
    int n;
    n = 20;
    for(i = 0; i < n; i-- )
        printf("-");
    return 0;
}
```

a) Forklar hva som er feil i programmet.

Løsningsforslag 3a:
for løkken teller nedover fremfor oppover (i-- burde være i++)

b) Det finnes tre måter å endre programmet slik at det fungerer som ment ved å legge til eller endre ***ett*** tegn. Foreslå ***en*** av disse endringene.

Løsningsforslag 3b:

Oppgaven viste seg å være vanskelig og den ble ikke vektlagt så mye under sensur. Tvilsomt at det vil dukke opp slike oppgaver på fremtidige eksamener.

1:
Endre 'i < n' til '-i < n'

2:
Endre 'i < n' til 'i + n'

3:
Endre 'i--' til 'n--'

Oppgave 4 - 35%

Denne oppgaven involverer bruk av lister og et angitt sett med listefunksjoner. Bruk de angitte listefunksjonene i besvarelsen. **Ikke** gjør antagelser om hvordan listene er implementert.

a) Skriv en funksjon som avgjør om en liste inneholder et bestemt element:

```
int list_contains(list_t *list, void *item)
```

list_contains skal returnere 1 dersom det angitte elementet (*item*) eksisterer i listen (*list*) og 0 dersom det ikke eksisterer. Her må du bruke listeiteratorer. Du kan anta at det eksisterer en funksjon *isequal* som avgjør om to elementer er like. *isequal* returnerer 1 dersom de to angitte elementene er like og 0 dersom de ikke er like:

```
int isequal(void *itemX, void *itemY)
```

Løsningsforslag 4a:

```
int list_contains(list_t *list, void *item)
{
    list_iterator_t *iter;
    void *tmp;

    iter = list_createiterator(list);
    tmp = list_next(iter);
    while (tmp != NULL) {
        if (isequal(tmp, item) == 1)
            break;
        tmp = list_next(iter);
    }
    list_destroyiterator(iter);
    if (tmp != NULL)
        return 1;
    else
        return 0;
}
```

b) Gitt en liste *A* og en liste *B*, skriv en funksjon som konstruerer en ny liste med de elementer som eksisterer i både *A* og *B*:

```
list_t *list_containsboth(list_t *A, list_t *B)
```

Her kan du gjenbruke funksjonen *list_contains* fra forrige oppgave.

Løsningsforslag 4b:

```
list_t *list_containsboth(list_t *A, list_t *B)
{
    list_t *new;
    list_iterator_t *iter;
    void *item;

    new = list_create();
    iter = list_createiterator(A);
    item = list_next(iter);
    while (item != NULL) {
        if (list_contains(B, item) == 1) {
            list_addfirst(new, item);
        }
        item = list_next(iter);
    }
    list_destroyiterator(iter);
    return new;
}
```

Du kan anta at følgende listefunksjoner er tilgjengelige.

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

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

// Lag en ny listeiterator
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 det ikke finnes noe neste element.
void *list_next(list_iterator_t *iter);

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