

# **SENSORVEILEDNING**

**For eksamen i: INF-1100**

---

**Dato: Mandag 25. februar 2019**

**Sensorveiledningen er på 7 sider inklusiv forside**

**Fagperson/intern sensor: Robert Pettersen**

**Telefon: 47 24 75 52**

**EksamensINF-1100  
Innføring i programmering og  
datamaskiners virkemåte  
Vår 2019  
Løsningsforslag**

*Eksamenssettet består av 4 oppgaver.*

Les oppgaveteksten grundig og disponer tiden slik at du får tid til å svare på alle oppgavene. I noen oppgaver kan det være nødvendig å tolke oppgaveteksten ved å gjøre noen antagelser - gjør i så fall rede for hvilke antagelser du har gjort, men pass på å ikke gjøre antagelser som trivialiserer oppgaven.

Der du skal utvikle eller beskrive en algoritme anbefales det at du først beskriver algoritmen på et høyt abstraksjonsnivå, f.eks. med figurer, før du går videre med detaljer og eventuell pseudokode. Dersom det spørres etter en implementasjon i C kreves det ikke 100% syntaktisk korrekt kode. Dersom det spørres etter pseudokode kan du kan også skrive ren C-kode om du ønsker.

Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

## Oppgave 1 - 25%

Gi en kort beskrivelse av von Neumann modellen og instruksjonssyklusen. Beskrivelsen bør omfatte de ulike komponentene i modellen og hvordan disse interagerer med hverandre.

---

### Løsningsforslag

---

Alle komponentene skal være kort beskrevet, helst med en tegning/figur. Input/output med eksempler på hva disse enhetene kan være. Kontrollenhet, med sine to registre; PC og IR, hva disse brukes til, og hvordan disse brukes. Gjerne beskrevet i instruksjonssyklusen. Prosesserings enheten med ALU og registre. Interaksjon med minne, hvordan man leser/skriver. Signaler, write enabled og lignende.

Alle stegene i instruksjons syklusen (Fetch, decode, evaluate address, fetch operand, execute, store result) skal være forklart med ett par setninger.

---

## Oppgave 2 - 25%

Gitt følgende program:

```
#include <stdio.h>

int main()
{
    int x = 256;
    char *y = "256";
    int z[] = { 2, 5, 6 };

    if (y[0] != z[0]) {
        printf("%d\n", x + z[2]);
    }
    else {
        printf("%d\n", x / z[0]);
    }
    return 0;
}
```

- a) Forklar hva som er forskjellen på variablene x, y, og z, med spesielt fokus på hvilken type variablene har og hvordan verdiene deres er representert i datamaskinens minne.
- b) Hvilket tall vil skrives ut på skjermen når programmet kjøres? Forklar hvordan du kommer frem til svaret ditt.

---

### Løsningsforslag

---

- a) x er av typen int (heltall), med verdien 256. x er en lokal variabel og ligger på stacken. y er en peker til et array av typen char (tekst streng) med 4 elementer; '2', '5', '6' og '\0' (0) som avslutter tekst-strengen. y ligger på stacken og holder kun minneadressen til selve elementene som ligger i dynamisk minne. z er et array av typen int (heltall) med 3 elementer; 2, 5 og 6. z er en lokal variabel som lagrer tallene sekvensielt på stacken.
- b) Programmet vil skrive ut integeret 262. If testen vil slå til, siden char verdien '2' ikke er lik integer verdien 2. I if clausen blir x addert med z[2] (256 + 6) som blir 262.

## Oppgave 3 - 25%

Gitt at en tekststreng er et array med elementer av type *char* og hvor siste element i arrayet har verdien 0.

- a) Skriv en funksjon *strlengde* returnerer antall elementer i en tekststreng, inkludert elementet med verdien 0:

```
int strlengde(char *s)
```

- b) Skriv en funksjon *strsiste* som returnerer arrayposisjonen til siste forekomst av et element *c* i en tekststreng *s*:

```
int strsiste(char *s, char c)
```

*strsiste* skal returnere verdien  $-1$  dersom dersom *c* ikke forekommer i *s*.

- c) Skriv en funksjon *strforekommer* som avgjør om en tekststreng *b* forekommer i en tekststreng *a*:

```
int strforekommer(char *a, char *b)
```

*strforekommer* skal returnere 1 dersom *b* forekommer i *a*.  $-1$  skal returneres dersom *b* ikke forekommer i *a*. For eksempel, tekststrengen "over" forekommer i tekststrengen "avisoverskrifter", men "gen" forekommer ikke i "formel".

---

### Løsningsforslag

---

- a) **int strlengde(char \*s)**  
{  
    **int** len;  
    **for** (len = 1; \*s != 0; s++)  
        len++;  
    **return** len;  
}
- b) **int strsiste(char \*s, char c)**  
{  
    **int** pos;  
    **int** siste;  
  
    siste = -1;  
    **for** (pos = 0; s[pos] != 0; pos++) {  
        **if** (s[pos] == c)  
            siste = pos;  
    }  
    **return** siste;  
}

```
c) int match(char *a, char *b)
{
    int i;

    for (i = 0; a[i] != 0 && b[i] != 0 && a[i] == b[i]; i++) ;

    if (b[i] == 0)
        return 0;
    return -1;
}

int strforekommer(char *a, char *b)
{
    int i;

    for (i = 0; a[i] != 0; i++) {
        if (match(a + i, b) == 0)
            return 1;
    }
    return -1;
}
```

---

## Oppgave 4 - 25%

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)
```

- 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.

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);
```

---

Løsningsforslag

---

a) `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) `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;  
}
```

---