

EKSAMENSOPPGAVE

Eksamen i:	INF-1100 - Innf. i progr. og datam. virkem.
Dato:	Tirsdag 5. desember 2017
Klokkeslett:	09:00-13:00
Sted:	Teorifagb., hus 3, 3218 og 3203
Tillatte hjelpemidler:	Ingen
Type innføringsark (rute/linje):	-
Antall sider inkl. forside:	5
Kontaktperson under eksamen:	Robert Pettersen
Telefon/mobil:	47247552
Vil det bli gått oppklaringsrunde i eksamenslokalet? Svar: <u>JA</u> / NEI	
Hvis JA: ca. kl. 11:00	

NB! Det er ikke tillatt å levere inn kladdepapir som del av eksamensbesvarelsen. Hvis det likevel leveres inn, vil kladdepapiret bli holdt tilbake og ikke bli sendt til sensur.

Eksamen INF-1100
Innføring i programmering og
datamaskiners virkemåte
Høst 2017

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 trivialisere 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 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.

Oppgave 2 - 25%

Gitt følgende funksjon:

```
int ukjent(int tall)
{
    int x = 0;

    while (tall != 0) {
        tall = tall >> 1;
        x++;
    }

    return x;
}
```

a) Hvilken verdi vil funksjonen *ukjent* ovenfor returnere ved følgende kall:

(1) *ukjent(3)*

(2) *ukjent(11)*

b) Hva er det funksjonen beregner?

Oppgave 3 - 25%

Gitt et array A med heltall (int i C) mellom 0 og 99.

- a) Skriv en funksjon `forekommer` som teller hvor mange ganger et gitt tall k forekommer i A . Funksjonen skal ta som inn-parametre: en peker til A , en angivelse av antall tall i A , samt en verdi for k :

```
int forekommer(int *A, int lengdeA, int k);
```

- b) Skriv en funksjon `frekvens` som teller hvor mange ganger tallene mellom 0 og 99 forekommer i A . Funksjonen skal skrive antall forekomster inn i et array B , slik at $B[0]$ inneholder antall forekomster av verdien 0; $B[1]$ inneholder antall forekomster av verdien 1, osv. Funksjonen skal ta som inn-parametre en peker til A , en angivelse av antall elementer i A , samt en peker til B :

```
void frekvens(int *A, int lengdeA, int *B);
```

Oppgave 4 - 25%

I denne oppgaven skal vi håndtere lister med ukjent antall elementer. Det eksisterer en funksjon `sammenlign` som tar to elementer som argument og returnerer en verdi som indikerer forholdet mellom dem:

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

Funksjonen returnerer verdien -1 dersom $e1$ er mindre enn $e2$, 0 dersom $e1$ er lik $e2$, og 1 dersom $e1$ er større enn $e2$.

- a) Gitt to lister a og b , begge sortert og med minste element først. Lag funksjonen `merge` som tar a og b som argument og returnerer en ny liste med elementene fra a og b sortert i rekkefølge, fra minst til størst:

```
list_t *merge(list_t *a, list_t *b);
```

- b) Gitt to lister a og b , begge sortert og med minste element først i listen, lag en funksjon `mergenum` som tar a og b som argument og returnerer antall unike elementer.

```
int mergenum(list_t *a, list_t *b);
```

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 sist i listen
void list_addlast(list_t *list, void *item);

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

// Returnerer elementet som pekes på av iteratoren og
// lar iteratoren peke på neste element.
// NULL returneres når en kommer til slutten av listen
void *list_next(list_iterator_t *iterator);

// Frigir iteratoren
void list_destroyiterator(list_iterator_t *iterator);
```