UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	**INF-1100 Innføring i programmering og datamaskiners virkemåte**
Dato:	**10.12.2019**
Klokkeslett:	**09:00 – 13:00**
Sted:	**ADM-B154 og ADM-K1.04**
Tillatte hjelpemidler:	**Ingen**
Type innføringsark (rute/linje):	**WiseFlow**
Antall sider inkl. forside:	**6**
Kontaktperson under eksamen: Telefon/mobil:	**Robert Pettersen** **47 24 75 52**

Vil det bli gått oppklaringsrunde i eksamenslokalet?

JA, ca. kl.: 10:30

NB! Det er ikke tillatt å levere inn kladdepapir som del av eksamensbesvarelsen.

UNIVERSITETET I TROMSØ

Eksamen INF-1100
Innføring i programmering og datamaskiners virkemåte
Høst 2019

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.

Oppgave 2 - 25%

Gitt følgende funksjon:

```
int ukjent(int a, int b)
{
    int x = 1;

    while(b > 0) {
        x *= a;
        b -= 1;
    }

    return x;
}
```

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

(1) *ukjent(2, 5)*

(2) *ukjent(10, 3)*

b) Hva er det funksjonen beregner?

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* som avgjør 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);
```

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

c) Skriv en funksjon *strrevers* som bytter om på elementene i en tekststreng slik at de står i motsatt rekkefølge. Elementet med verdien '\0' skal ikke bytte plass.

```
void strrevers(char *s);
```

Oppgave 4 - 25%

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

a) Skriv en funksjon som reverserer rekkefølgen på elementene i en liste:

```
void list_reverse(list_t *list);
```

Funksjonen skal ikke returnere en ny liste, men elementene i listen skal endre rekkefølge slik at de ligger i reversert rekkefølge.

b) 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 listeiterator. 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)
```

c) 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
void list_addfirst(list_t *list, void *item);

// Fjerner et element fra listen
void list_remove(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);
```

UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGÅVE

Eksamen i:	**INF-1100 Innføring i programmering og datamaskiner sin virkemåte**
Dato:	**10.12.2019**
Klokkeslett:	**09:00 – 13:00**
Stad:	**ADM-B154 og ADM-K1.04**
Lovlege hjelpemiddel:	**Ingen**
Type innføringsark (rute/linje):	**WiseFlow**
Antall sider inkl. framside:	**6**
Kontaktperson under eksamen: Telefon/mobil:	**Robert Pettersen** **47 24 75 52**
Skal det gåast oppklarande runde i eksamenslokalet? **JA, ca. kl.: 10:30**	

NB! Det er ikkje lov å levere inn kladd saman med svaret.

Om den likevel vert levert inn, vil kladden verte halden attende og ikkje sendt til sensur.

UNIVERSITY OF TROMSØ

Eksamen INF-1100
Innføring i programmering og datamaskiner sin virkemåte
Haust 2019

Eksamenssettet består av 4 oppgåver.

Les oppgåveteksten grundig og disponer tida slik at du får tid til å svara på alle oppgåvene. I nokre oppgåver kan det vera naudsynt å tolka oppgåveteksten ved å gjera nokre føresetnader (antakelser) - gjer i så fall greie for kva slags føresetnader du har gjort, men pass på å ikkje gjera føresetnader som trivialiserer oppgåva.

Der du skal utvikla eller skildra ein algoritme tilrådast det at du først skildrar algoritmen på eit høgt abstraksjonsnivå, t.d. med figurar, før du går vidare med detaljar og eventuell pseudokode. Dersom det vert spurt etter ein implementasjon i C krev det ikkje 100% syntaktisk korrekt kode. Dersom det vert spurt etter pseudokode kan du òg skriva rein C-kode om du ynskjer.

Husk òg at du kan referera attende til funksjonar du tidlegare har definert.

Oppgåve 1 - 25%

Gje ei kort skildring av von Neumann modellen og instruksjonssyklusen. Skildringa bør omfatta dei ulike komponentane i modellen og korleis desse interagerer med kvarandre.

Oppgåve 2 - 25%

Gitt følgjande funksjon:

```
int ukjent(int a, int b)
{
    int x = 1;

    while(b > 0) {
        x *= a;
        b -= 1;
    }

    return x;
}
```

a) Kva for ein verdi vil funksjonen *ukjent* ovanfor returnera ved følgjande kall:

(1) *ukjent(2, 5)*

(2) *ukjent(10, 3)*

b) Kva er det funksjonen bereknar?

Oppgåve 3 - 25%

Gitt at ein tekststreng er eit array med element av type *char* og kor siste element i arrayet har verdien'\0'.

a) Skriv ein funksjon *strlengde* som avgjer talet på element i ein tekststreng, inkludert elementet med verdien '\0'.

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

b) Skriv ein funksjon *strsiste* som returnerer arrayposisjonen til siste førekomst av eit element *c* i ein tekststreng *s*.

```
int strsiste(char *s);
```

strsiste skal returnera verdien -1 dersom element *c* ikkje finst i tekststrengen *s*.

c) Skriv ein funksjon *strrevers* som byter om på elementa i ein tekststreng slik at dei står i motsett rekkefølgje. Elementet med verdien '\0' skal ikkje byta plass.

```
void strrevers(char *s);
```

Oppgåve 4 - 25%

Denne oppgåva involverer bruk av lister og eit angitt sett med listefunksjonar. Bruk dei angitte listefunksjonane på neste side. **Ikkje** gjer antagelser om korleis listene er implementerte.

a) Skriv ein funksjon som reverserer rekkefølgja på elementa i ei liste:

```
void list_reverse(list_t *list);
```

Funksjonen skal ikkje returnera ei ny liste, men elementa i lista skal endra rekkefølgje slik at dei ligg i reversert rekkefølgje.

b) Skriv ein funksjon som avgjer om ei liste inneheld eit bestemt element:

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

list_contains skal returnera 1 dersom det angitte elementet (*item*) eksisterer i lista (*list*) og 0 dersom det ikkje eksisterer. Her må du bruka listeiterator. Du kan anta at det eksisterer ein funksjon *isequal* som avgjer om to element er like. *isequal* returnerer 1 dersom dei to angitte elementa er like og 0 dersom dei ikkje er like:

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

c) Gitt ei liste *A* og ei liste *B*, skriv ein funksjon som konstruerer ei ny liste med dei elementa som eksisterer i både *A* og *B*:

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

Her kan du gjenbruke funksjonen *list_contains* frå førre oppgåve.

Du kan anta at følgjande listefunksjonar er tilgjengelege:

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

// Set inn eit element først i ei liste
void list_addfirst(list_t *list, void *item);

// Fjernar eit element frå lista
void list_remove(list_t *list, void *item);

// Lag ein ny listeiterator
list_iterator_t *list_createiterator(list_t *list);

// Returner element som blir peika på av iterator og
// la iterator peika på neste element. NULL blir returnert
// når det ikkje finst noko neste element.
void *list_next(list_iterator_t *iter);

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