UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	INF-1100 – Innf. i progr. og datam. virkem.
Dato:	05.12.2018
Klokkeslett:	09:00 – 13:00
Sted:	Kraft I og II Hall del 3
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? Svar: JA / NEI Hvis JA: ca. kl.__10:30_____	

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 2018

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 = 0;

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

    return x;
}
```

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

(1) *ukjent(12, 5)*

(2) *ukjent(16, 3)*

b) Hva er det funksjonen beregner?

Oppgave 3 - 25%

Gitt et array *A* som inneholder n heltall (integers).

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 n, int k);
```

b) Skriv en funksjon `sorter` som flytter rundt på tallene i A slik at de blir liggende i stigende rekkefølge. Funksjonen skal ta som inn-parametre: en peker til A og en angivelse av antall tall i A. Funksjonen skal ikke returnere noe.

```
void sorter(int *A, int n);
```

c) A har ett *majoritetselement* dersom ett tall forekommer mer enn $n/2$ ganger i A. Følgende algoritme kan benyttes for å avgjøre om A har ett majoritetselement. Anta at vi først sorterer A. Dersom A har ett majoritetselement må dette tallet være i $A[n/2]$. Dersom antall forekomster av $A[n/2]$ i A er høyere enn $n/2$, er $A[n/2]$ ett marjoritetselement.

Skriv en funksjon som avgjør om A har ett majoritetselement. Funksjonen skal ta som inn-parametre: en peker til A og en angivelse av antall tall i A. Funksjonen skal returnere 1 dersom det finnes ett majoritetselement, og 0 dersom det ikke finnes.

Bruk gjerne funksjonene du har implementert tidligere.

```
int majoritetselement(int *A, int n);
```

Oppgave 4 - 25%

I ett orienteringsløp skal en deltaker besøke ett gitt antall poster i en gitt rekkefølge. På hver post vil ett kontrollelement som er unik for posten bli lagd når deltakeren er innom posten. Dette kontrollelementet inneholder en unik kode for posten (*kode*) og hvor lang tid deltakeren har brukt fra forrige post, eventuelt fra start hvis dette er første post (*strekktid*):

```
typedef struct kontrollelement kontrollelement_t;
struct kontrollelement {
    int kode;
    double strekktid;
};
```

Kontrollelementet legges til deltakerens liste over poster som er besøkt (*spor*). Mål håndteres på samme måte som alle andre poster (men skal være siste post som registreres). Når deltakerne kommer i mål vil listen av kontrollelementer kunne kontrolleres mot en fasit-liste hvor postene er plassert i rett rekkefølge (*fasit*).

a) Lag en funksjon som kontrollerer at deltakeren har besøkt alle postene, og at de er besøkt i rett rekkefølge. Funksjonen skal returnere 1 hvis deltakeren har besøkt alle postene i rett rekkefølge, 0 hvis dette ikke er tilfelle og -1 hvis noe går galt under sjekken:

```
int kontroll(list_t *spor, list_t *fasit);
```

b) Lag en funksjon som returnerer tiden deltakeren har brukt fra start til mål. Returner -1 dersom noe skulle gå galt under beregningen:

```
double tid(list_t *spor);
```

c) For alle deltakerne som har besøkt alle postene i rett rekkefølge tar vi vare på deres spor i en ny liste *godkjente* (en liste hvor elementene er lister tilsvarende *spor* ovenfor). Lag en funksjon som tar *godkjente* som argument, og finner den deltakeren med beste sammenlagt tid (minste sammenlagte tid). Funksjonen skal returnere denne bestetiden, eller -1 hvis noe går galt under beregningen:

```
double bestetid(list_t *godkjente);
```

(Hint: Du kan benytte funksjonen *tid* fra forrige deloppgave.)

Ikke gjør noen antakelser om hvordan listen er implementert. Du kan anta at følgende listefunksjoner er tilgjengelige:

```
// 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);
```

UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGÅVE

Eksamen i:	INF-1100 – Innf. i progr. og datam. virkem.
Dato:	05.12.2018
Klokkeslett:	09:00 – 13:00
Stad:	Kraft I og II Hall del 3
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? Svar: JA / Nei Dersom 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.

Eksamen INF-1100
Innføring i programmering og datamaskiner sin virkemåte
Haust 2018

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%

Gjeve følgjande funksjon:

```
int ukjent(int a, int b)
{
    int x = 0;

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

    return x;
}
```

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

(1) *ukjent(12, 5)*

(2) *ukjent(16, 3)*

b) Kva er det funksjonen bereknar?

Oppgåve 3 - 25%

Gjeve eit array A som inneheld n heiltal (integers).

a) Skriv ein funksjon `finst` som tel kor mange gonger eit gjeve tal k finst i A. Funksjonen skal ta som inn-parametrar: ein peikar til A, ein angivelse av mengd tal i A, og dessutan ein verdi for k:

```
int finst(int *A, int n, int k);
```

b) Skriv ein funksjon `sortar` som flyttar rundt på tala i A slik at dei vert liggjande i stigande rekkjefølgje. Funksjonen skal ta som inn-parametrar: ein peikar til A og ein angivelse av mengd tal i A. Funksjonen skal ikkje returnera noko.

```
void sortar(int *A, int n);
```

c) A har eitt *majoritetselement* dersom eitt tal finst meir enn $n/2$ gonger i A. Følgjande algoritme kan nyttast for å avgjera om A har eitt majoritetselement. Anta at vi først sorterer A. Dersom A har eitt majoritetselement må dette talet vera i $A[n/2]$. Dersom mengd førekomstar av $A[n/2]$ i A er høgare enn $n/2$, er $A[n/2]$ eitt marjoritetselement.

Skriv ein funksjon som avgjer om A har eitt majoritetselement. Funksjonen skal ta som inn-parametrar: ein peikar til A og ein angivelse av mengd tal i A. Funksjonen skal returnera 1 dersom det finst eitt majoritetselement, og 0 dersom det ikkje finst.

Bruk gjerne funksjonane du har implementert tidlegare.

```
int majoritetselement(int *A, int n);
```

Oppgåve 4 - 25%

I eitt orienteringsløp skal ein deltakar vitja éi gjeve mengd postar i ei gjeven rekkjefølgje. På kvar post vil eitt kontrollelement som er unik for posten lagast når deltakaren er innom posten. Dette kontrollelementet inneheld ein unik kode for posten (*kode*) og kor lang tid deltakaren har brukt frå førre post, eventuelt frå start viss dette er første post (*strekktid*):

```
typedef struct kontrollelement kontrollelement_t;
struct kontrollelement {
    int kode;
    double strekktid;
};
```

Kontrollelementet vert lagt til lista til deltakaren over postar som er vitja (*spor*). Mål vert handtert på same måte som alle andre postar (men skal vera siste post som vert registrert). Når deltakarane kjem i mål vil lista av kontrollelement kunna kontrollerast mot ein fasit-liste der postane er plassert i rett rekkjefølgje (*fasit*).

a) Lag ein funksjon som kontrollerer at deltakaren har vitja alle postane, og at dei er vitja i rett rekkjefølgje. Funksjonen skal returnera 1 viss deltakaren har vitja alle postane i rett rekkjefølgje, 0 viss dette ikkje er tilfelle og -1 viss noko går gale under sjekken:

```
int kontroll(list_t *spor, list_t *fasit);
```

b) Lag ein funksjon som returnerer tida deltakaren har brukt frå start til mål. Returner -1 dersom noko skulle gå gale under beregningen:

```
double tid(list_t *spor);
```

c) For alle deltakarane som har vitja alle postane i rett rekkjefølgje tek vi vare på sporet deira i ei ny liste *godkjende* (ei liste der elementa er lister tilsvarande *spor* ovanfor). Lag ein funksjon som tek *godkjende* som argument, og finn den deltakaren med beste samanlagt tid (minste samanlagte tid). Funksjonen skal returnera denne bestetiden, eller -1 viss noko går gale under beregningen:

```
double bestetid(list_t *godkjende);
```

(Hint: Du kan nytta funksjonen *tid* frå førre deloppgave.)

Ikkje gjer nokon antakelser om korleis lista er implementert. Du kan anta at følgjande listefunksjonar er tilgjengelege:

```
// Lag ei ny liste-iterator som peikar på første element i lista
list_iterator_t *list_createiterator(list_t *list);

// Returnerer elementet som vert peikt på av iteratoren og
// let iteratoren peika på neste element.
// NULL vert returneres når ein hjem til slutten av lista.
void *list_next(list_iterator_t *iterator);

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