EKSAMENSOPPGAVE I INF-1100

Eksamen i	:	**Innføring i programmering**
Eksamensdato	:	**24 februar**
Tid	:	**0900-1300**
Sted	:	**B154**
Tillatte hjelpemidler	:	**Ingen**

Oppgavesettet er på 4 sider ekskl. forside

Kontaktperson under eksamen: Steffen Valvåg, tlf 48011116

FAKULTET FOR NATURVITENSKAP OG TEKNOLOGI
Universitetet i Tromsø

Eksamen INF-1100
Innføring i programmering
Vår 2011

Eksamenssettet består av 4 oppgaver.

Der oppgaven ber om at du skriver en funksjon kan du bruke C lignende pseudo-kode. Husk også at du kan referere tilbake til funksjoner du tidligere har definert.

Oppgave 1 - 25%

a) Beskriv komponentene i von Neumann modellen.

b) Beskriv de ulike fasene i en instruksjonssyklus.

Oppgave 2 - 25%

Gitt ett array `A` med heltall 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`:

```
void 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 til 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 tall i `A`, samt en peker til `B`:

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

Oppgave 3 - 25%

Gitt ett array A og B med tilfeldige heltall.

a) Skriv en funksjon *overlapp* som avgjør hvilke tall som finnes både i `A` og `B`. Funksjonen skal skrive disse tallene til et array `C`. Funksjonen skal ta som inn-parametre en peker til `A`, en angivelse av antall tall i `A`, en peker til `B`, en angivelse av antall tall i `B`, samt en peker til `C`:

```
void overlapp(int *A, int lengdeA, int *B, int lengdeB, int *C);
```

b) Skriv en funksjon *overlappsortert* som utfører samme oppgave som funksjonen *overlapp* i **a)**, men organiserer tallene i `C` slik at de er sortert i stigende rekkefølge.

Oppgave 4 - 25%

Gitt en liste `bokmål` med bokmålsord og en liste `nynorsk` med nynorske ord. Hvert ord er beskrevet som en tekststreng.

a) Skriv en funksjon *strcmp* som avgjør om to tekststrenger er like. Funksjonen skal ta som inn-parametre en peker til en tekststreng `a` og en tekststreng `b`. Funksjonen skal returnere 0 dersom `a` og `b` inneholder samme tegn i samme rekkefølge, og 1 hvis ikke:

```
int strcmp(char *a, char *b);
```

Merk at `a` og `b` er **ikke** like dersom de har ulik lengde. Du kan anta at tekststrengene er avsluttet med verdien 0.

b) Skriv en funksjon *listeoverlapp* som avgjør hvilke ord som forekommer **kun i den ene** av `bokmål` og `nynorsk` listene. Funksjonen skal fjerne disse ordene fra sin tilhørende liste, plassere disse i en ny liste, og returnere en peker til den nye listen. Funksjonen skal ta som inn-parametre en peker til en liste `bokmål` og en peker til en liste `nynorsk`:

```
list_t *listeoverlapp(list_t *bokmål, list_t *nynorsk);
```

For eksempel, om `bokmål` listen inneholder ordene `{"bok", "tilbake", "hvordan"}` og `nynorsk` listen inneholder ordene `{"korleis", "attende", "bok"}` skal funksjonen returnere en liste som inneholder ordene `{"tilbake", "hvordan", "korleis", "attende"}`.

Du kan anta at følgende listefunksjoner er tilgjengelige:

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

// Sett inn et element sist i en liste
void list_addlast(list_t *list, void *item);

// Fjern et element fra en liste
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
void *list_next(list_iterator_t *iter);

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