EKSAMENSOPPGAVE I INF-1100

Eksamen i: INF-1100 Innføring i programmering og datamaskiners virkemåte

Eksamensdato: Mandag 12. Desember 2011

Tid: Kl. 09:00 – 13:00

Sted: Åsgårdvegen 9

Tillatte hjelpemidler: Ingen

Oppgavesettet er på 4 sider eksklusiv forside

Kontaktperson under eksamen: Åge Kvalnes

Telefon: 93030504

FAKULTET FOR NATURVITENSKAP OG TEKNOLOGI
Universitetet i Tromsø

Eksamen INF-1100
Innføring i programmering og datamaskiners virkemåte
Høst 2011

Eksamenssettet består av 4 oppgaver.

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

Oppgave 1 - 20%

Anta at symbolet ^ representerer den logiske operasjonen *eksklusiv or*. Følgende tabell viser resultatet av ^ operasjonen mellom to binære siffer:

```
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0
```

Gitt to variabler x og y. Anta at x har verdien 3 og y verdien 5. Hvilke verdier har x og y **etter** at følgende sekvens av operasjoner er utført?

```
x = x ^ y
y = x ^ y
x = x ^ y
```

Oppgave 2 - 20%

De fleste av dagens datamaskiner er strukturert i henhold til en modell foreslått av John von Neumann i 1946.

a) Gi en kort beskrivelse av komponentene i von Neumann modellen.

b) Beskriv kort hvordan I/O utføres i en von Neumann-basert datamaskin.

Oppgave 3 - 30%

Gitt et array A som inneholder n heltall.

(a) Skisser i pseudo-kode en funksjon som organiserer tallene i A slik at oddetall kommer før partall.

(b) Skisser i pseudo-kode en funksjon som sorterer tallene i A i stigende rekkefølge.

(c) A har et *majoritetselement* dersom et tall forekommer mer enn $n/2$ ganger i A.

Følgende algoritme kan benyttes for å avgjøre om A har et majoritetselement. Anta at vi først sorterer A. Dersom A har et majoritetselement må dette tallet være i $A[n/2]$. Dersom antall forekomster av $A[n/2]$ i A er høyere enn $n/2$, er $A[n/2]$ et majoritetselement.

Skisser i pseudo-kode en funksjon som avgjør om A har et majoritetselement. Her kan du gjenbruke sorteringsfunksjonen fra b).

Oppgave 4 - 20%

(a) Skisser i pseudo-kode en funksjon som tar en liste som argument og returnerer en ny liste med de samme elementene som i argument-listen, men i reversert rekkefølge.

(b) Skisser i pseudo-kode en funksjon som finner det k **siste** element i en liste. For eksempel, dersom listen har 10 elementer og k er lik 3, skal element nummer 7 returneres. Du kan anta at k har en verdi mindre eller lik lengden på listen.

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
int list_addlast(list_t *list, void *item);

// Sett inn et element først i en liste
int list_addfirst(list_t *list, void *item);

// Lag en ny listeiterator som peker på første element i listen
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 en når slutten på listen.
void *list_next(list_iterator_t *iter);

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

EKSAMENSOPPGÅVE I INF-1100

Eksamen i: INF-1100 Innføring i programmering og datamaskiner sin virkemåte

Eksamensdato: Mandag 12. Desember 2011

Tid: Kl. 09:00 – 13:00

Sted: Åsgårdvegen 9

Tillate hjelpemidler: Ingen

Oppgåvesettet er på 4 sider eksklusiv forside

Kontaktperson under eksamen: Åge Kvalnes

Telefon: 93030504

FAKULTET FOR NATURVITENSKAP OG TEKNOLOGI
Universitetet i Tromsø

Eksamen INF-1100
Innføring i programmering og datamaskiner sin virkemåte
Haust 2011

Eksamenssettet består av 4 oppgåver.

Der oppgåva ber om at du skriv ein funksjon kan du bruke C liknande pseudokode. Hugs og at du kan referere attende til funksjonar du tidlegare har definert.

Oppgåve 1 - 20%

Gå ut frå at symbolet ^ representerer den logiske operasjonen *eksklusiv or*. Følgjande tabell viser resultatet av ^ operasjonen mellom to binære siffer:

```
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0
```

Gjeve to variablar x og y. Gitt at x har verdi 3 og y verdi 5. Kva for verdi har x og y etter at følgjende sekvens av operasjonar er utførde.

```
x = x ^ y
y = x ^ y
x = x ^ y
```

Oppgåve 2 - 20%

Dei fleste av dagens datamaskiner er strukturert i samhøve med ein modell føreslått av John von Neumann i 1946.

a) Gje ei kort skildring av komponentane i von Neumann modellen.

c) Skildre kort korleis I/O vert utført i ein von Neumann-basert datamaskin.

Oppgåve 3 - 30%

Gjeven eit array A med n heiltal.

(a) Skisser i pseudo-kode ein funksjon som organiserer tala i A slik at oddetal kjem før partal.

(b) Skisser i pseudo-kode ein funksjon som sorterer tala i A i stigande rekkefølgje.

(c) A har eit *majoritetselement* dersom eit tal finst meir enn $n/2$ gonger i A.

Følgjande algoritme kan nyttast for å avgjere om A har eit majoritetselement. Gå ut frå at ein først sorterer A. Om A har eit majoritetselement må dette talet vere i $A[n/2]$. Om mengd førekomstar av $A[n/2]$ i A er større enn $n/2$, er $A[n/2]$ eit majoritetselement.

Skisser i pseudo-kode ein funksjon som avgjer om A har et majoritetselement. Her kan du nytte sorteringsfunksjonen frå b).

Oppgåve 4 - 20%

(a) Skisser i pseudo-kode ein funksjon som tek ei liste som argument og returnerer ei ny liste med dei same elementa som i argument-lista, men i reversert rekkefølgje.

(b) Skisser i pseudo-kode en funksjon som finn det k **siste** element i ei liste. Til dømes, om lista har 10 element og k er lik 3, skal element nummer 7 returnerast. Du kan gå ut frå at k har en verdi mindre eller lik lengda på lista.

Du kan gå ut frå at følgjande listefunksjonar er tilgjengelege:

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

// Sett inn eit element sist i ei liste
int list_addlast(list_t *list, void *item);

// Sett inn eit element fyrst i ei liste
int list_addfirst(list_t *list, void *item);

// Lag ein ny listeiterator som peikar på fyrste element i lista
list_iterator_t *list_createiterator(list_t *list);

// Returner element som vert peika på av iterator og
// la iterator peike på neste element.  NULL
// vert returnert når ein når slutten på lista.
void *list_next(list_iterator_t *iter);

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