UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i: **INF-1100**
Innføring i programmering og datamaskiners virkemåte
Dato: **Tirsdag 8. desember 2015**
Tid: **Kl 09:00 – 13:00**
Sted: **Teorifagbygget, Hus 1**

Tillatte hjelpemidler: **Ingen**

Oppgavesettet er på 11 sider inklusiv forside
(Bokmål side 2-6, Nynorsk side 7-11)

Kontaktperson under eksamen: Steffen Viken Valvåg
Telefon: 98 11 77 49

NB! Det er ikke tillatt å levere inn kladd sammen med besvarelsen

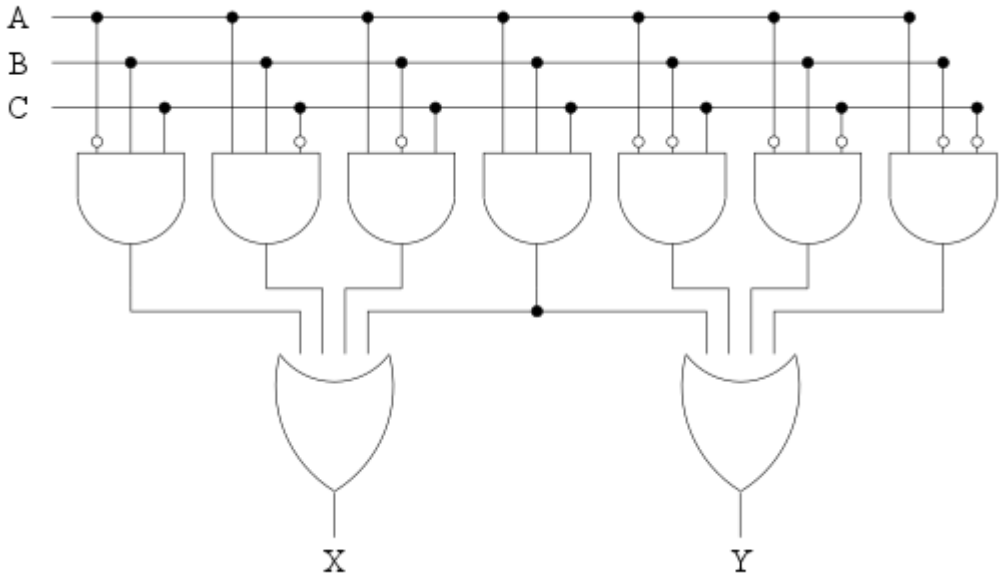
Les oppgaveteksten grundig og disponer tiden slik at du får tid til å svare på alle oppgavene. I noen oppgaver kan det være nødvendig å tolke oppgaveteksten, dersom noe er uklart. Beskriv i så fall hvilke antagelser du har gjort, men pass på at du ikke trivialiserer oppgaven.

Det er ønskelig at du skriver kode som ligger nærmest mulig opp mot gyldig C-kode. Dersom det spørres etter pseudokode kan du også skrive ren C-kode om du ønsker.

Husk at funksjoner som er implementert i en oppgave også kan brukes i andre oppgaver.

Oppgave 1 (20%)

Figuren under viser en logisk krets med tre binære input-verdier, A, B og C, og to binære output-verdier, X og Y.



a) Lag en tabell som viser verdien av X og Y for alle mulige kombinasjoner av input-verdier A, B, og C. (Det er 8 mulige kombinasjoner av input-verdier, så tabellen vil ha 8 rader.)

b) Hva ser ut til å være formålet med denne kretsen? Kan du gi en enkel forklaring eller en huskeregel for hva verdiene til X og Y er, basert på A, B, og C, slik at man ikke trenger å memorisere hele tabellen?

Oppgave 2 (15%)

a) I en prosessor, hva er et register?

b) I Von Neumann modellen er det en komponent som heter kontrollenhet (control unit) hvor det er et spesielt register som heter programteller (program counter). Hva er formålet med dette registeret, og hvordan oppdateres det?

c) I Von Neumann modellen er det en annen komponent som heter minne (memory). Denne komponenten inneholder to registere som heter MAR og MDR. Hva brukes disse registerene til?

Oppgave 3 (10%)

Euklids algoritme kan brukes til å beregne største felles divisor for to heltall A og B, altså det største tallet som deler begge tallene. I pseudokode kan algoritmen uttrykkes slik:

```
funksjon GCD(A, B):
    så lenge B er ulik 0:
        TMP settes lik B
        B settes lik A % B
        A settes lik TMP
    returner A
```

Oversett denne pseudokoden til C-kode, hvor du definerer en funksjon som heter **GCD** og hvor **A** og **B** er av typen **int**.

Oppgave 4 (10%)

Du ønsker nå å lage et program som kan regne med brøker, slik at du kan evaluere aritmetiske uttrykk helt eksakt, uten avrundingsfeil.

Første steg er å lage en datatype for rasjonelle tall (brøker). Et rasjonelt tall kan skrives som en brøk med et heltall som teller (over brøkstreken) og også et heltall som nevner (under brøkstreken).

Du har definert følgende datatype for dette formålet:

```
struct R {
    int teller;
    int nevner;
};
typedef struct R R;
```

Et problem som oppstår er at mange brøker er ekvivalente, men du foretrekker å bruke den mest reduserte formen av en brøk. I stedet for $\frac{14}{21}$ vil du altså foretrekke $\frac{2}{3}$. For å forenkle (også kalt redusere) en brøk, kan man regne ut største felles divisor for teller og nevner i brøken, og så dele begge på denne. For eksempel er $GCD(14, 21) = 7$, og hvis man i brøken $\frac{14}{21}$ deler både teller og nevner på 7 får man $\frac{2}{3}$.

Negative brøker vil du representere ved at telleren er negativ, mens nevneren er positiv.

Lag en funksjon som reduserer en brøk mest mulig, og i tillegg sikrer at nevneren i brøken aldri er et negativt tall. (Hvis nevneren er negativ, må både teller og nevner skifte fortegn.) Funksjonen skal ha følgende signatur:

```
void reduser(R *r);
```

Bruk funksjonen fra oppgave 3 ved behov.

Oppgave 5 (10%)

Du trenger et sett med funksjoner for å manipulere rasjonelle tall, så du kan gjøre addisjon og multiplikasjon.

Når man adderer to brøker $\frac{a}{b}$ og $\frac{c}{d}$ må man finne en felles nevner. Det enkleste er å bruke $b * d$ som felles nevner, slik at man ender opp med:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$$

For å multiplisere to brøker bruker man formelen:

$$\frac{a}{b} * \frac{c}{d} = \frac{ac}{bd}$$

Implementer følgende to funksjoner for å gjøre aritmetiske operasjoner på rasjonelle tall.

```
void add(R *r, R *x, R *y);
void mul(R *r, R *x, R *y);
```

Funksjonene skal beregne henholdsvis summen og produktet av tallene som **x** og **y** peker på, og tilordne resultatet til tallet som **r** peker på. Funksjonene skal også bruke funksjonen fra oppgave 4 til å forenkle resultatet mest mulig.

Oppgave 6 (10%)

Du skal nå implementere subtraksjon og divisjon for rasjonelle tall, basert på funksjonene fra oppgave 5. Implementer følgende to funksjoner:

```
void sub(R *r, R *x, R *y);
void div(R *r, R *x, R *y);
```

Funksjonen **sub** skal implementeres ved å kalle **add**-funksjonen med motsatt fortegn på **y**. Gjør dette på en slik måte at du ikke endrer verdien til **y** som en sideeffekt av å kalle **sub**.

Funksjonen **div** skal implementeres ved å kalle **mul**-funksjonen med den inverse verdien til **y**. (Dvs. at teller og nevner i **y** skal bytte plass.) Pass også her på at du ikke endrer verdien til **y** som en sideeffekt av å kalle **div**.

Oppgave 7 (25%)

Du vil representere en sekvens av rasjonelle tall som ei lenket liste. Du har definert følgende datatype for å representere en node i ei slik liste:

```
struct node {
    R verdi;
    struct node *neste;
};
typedef struct node node_t;
```

Implementer følgende fire funksjoner for å manipulere slike lister:

```
int lengde(node_t *liste);
void sum(R *r, node_t *liste);
void gjennomsnitt(R *r, node_t *liste);
void minimum(R *r, node_t *liste);
```

Funksjonen **lengde** skal returnere lengden til ei lenket liste, altså antall noder i lista. Du kan anta at lista termineres med en NULL-peker.

Funksjonen **sum** skal beregne summen av de rasjonelle tallene som ligger i ei liste. Resultatet skal tilordnes det tallet som **r** peker på.

Funksjonen **gjennomsnitt** skal beregne gjennomsnittsverdien av de rasjonelle tallene som ligger i ei liste. Resultatet skal tilordnes det tallet som **r** peker på.

Funksjonen **minimum** skal finne det minste rasjonelle tallet i ei liste og tilordne resultatet til det tallet som **r** peker på.

For tomme lister skal **lengde** returnere 0, og de andre funksjonene skal returnere uten å ha noen effekt, dvs. uten å endre verdien som **r** peker på.

For å implementere **mimimum** må du kunne sammenligne brøker. Merk at dersom $\frac{a}{b} < \frac{c}{d}$ så vet vi også at $a * d < b * c$.

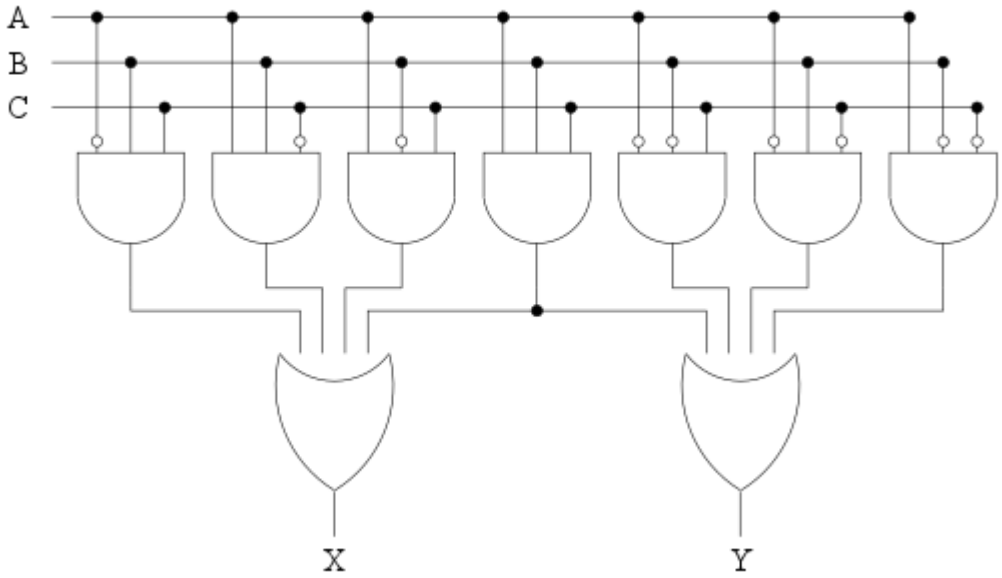
Les oppgåveteksten grundig og disponer tida så du får tid til å svare på alle oppgåvene. I somme oppgåver kan det vere nødvendig å tolke oppgåveteksten, dersom noko er uklart. Beskriv i så fall kva du har antatt, men pass på at du ikkje trivialiserer oppgåva.

Det er ønskeleg at du skriv kode som ligg nærast mogleg opp mot gyldig C-kode. Dersom oppgåva spør etter pseudokode kan du også skrive rein C-kode om du vil.

Husk at funksjonar som er implementert i ei oppgåve også kan bli brukt i andre oppgåver.

Oppgåve 1 (20%)

Figuren under viser ein logisk krets med tre binære input-verdiar, A, B og C, og to binære output-verdiar, X og Y.



a) Lag ein tabell som viser verdien av X og Y for alle moglege kombinasjonar av input-verdiar A, B, og C. (Det er 8 moglege kombinasjonar av input-verdiar, så tabellen vil ha 8 rader.)

b) Kva sjår ut til å vere formålet med denne kretsen? Kan du gi ei enkel forklaring eller ein huskeregel for kva verdiane til X og Y er, basert på A, B, og C, så du ikkje treng å memorisere heile tabellen?

Oppgåve 2 (15%)

a) I ein prosessor, kva er eit register?

b) I Von Neumann modellen er det ein komponent som heiter kontrolleining (control unit) som inneheld eit spesielt register som heiter programteljar (program counter). Kva er formålet med dette registeret, og korleis endrar det verdi?

c) I Von Neumann modellen er det også ein komponent som heiter minne (memory). Denne komponenten inneheld to register som heiter MAR og MDR. Kva er formålet med desse registera?

Oppgåve 3 (10%)

Euklid sin algoritme kan verte brukt til å rekne ut største felles divisor for to heiltal A og B, altså det største talet som deler begge tala. I pseudokode kan algoritmen sjå slik ut:

```
funksjon GCD(A, B):
    så lenge B er ulik 0:
        TMP settes lik B
        B settes lik A % B
        A settes lik TMP
    returner A
```

Oversett denne pseudokoden til C-kode, der du definerer ein funksjon som heiter **GCD** og der **A** og **B** har typen **int**.

Oppgåve 4 (10%)

Du ønsker no å lage eit program som kan rekne med brøkar, så du kan evaluere aritmetiske uttrykk heilt eksakt, utan å runde av.

Første steg er å lage ein datatype for rasjonelle tal (brøkar). Eit rasjonelt tal kan du skrive som ein brøk med eit heiltal som teljar (over brøkstreken) og også eit heiltal som nemnar (under brøkstreken).

Du har definert ein datatype for dette formålet:

```
struct R {
    int teller;  // teljar
    int nevner;  // nemnar
};
typedef struct R R;
```

Eit problem er at mange brøkar er ekvivalente, men du føretrekk å bruke den mest reduserte formen av ein brøk. I staden for $\frac{14}{21}$ vil du altså føretrekke $\frac{2}{3}$. For å forenkle (også kalt redusere) ein brøk, kan du rekne ut største felles divisor for teljar og nemnar i brøken, og så dele begge på denne. Til dømes er $GCD(14, 21) = 7$, og dersom du i brøken $\frac{14}{21}$ deler både teljar og nemnar på 7 får du $\frac{2}{3}$.

Negative brøkar vil du representere ved at teljaren er negativ, samstundes som nemnaren er positiv.

Lag ein funksjon som reduserer ein brøk mest mogleg, og i tillegg sjår til at nemnaren i brøken aldri er eit negativt tal. (Dersom nemnaren er negativ, må både teljar og nemnar skifte forteikn.) Funksjonen skal ha følgande signatur:

```
void reduser(R *r);
```

Bruk funksjonen frå oppgåve 3 der det trengst.

Oppgåve 5 (10%)

Du treng eit sett med funksjonar for å manipulere rasjonelle tal, så du kan gjere addisjon og multiplikasjon.

Når du adderer to brøkar $\frac{a}{b}$ og $\frac{c}{d}$ må du finne ein felles nemnar. Det enklaste er å bruke $b * d$ som felles nemnar, slik at du ender opp med:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$$

For å multiplisere to brøkar bruker du formelen:

$$\frac{a}{b} * \frac{c}{d} = \frac{ac}{bd}$$

Implementer følgande to funksjonar for å gjere aritmetiske operasjonar på rasjonelle tal.

```
void add(R *r, R *x, R *y);
void mul(R *r, R *x, R *y);
```

Funksjonane skal rekne ut summen og produktet, respektivt, av tala som **x** og **y** peiker på, og skrive resultatet til talet som **r** peiker på. Funksjonane skal også bruke funksjonen frå oppgåve 4 til å forenkle resultatet mest mogleg.

Oppgåve 6 (10%)

Du skal no implementere subtraksjon og divisjon for rasjonelle tal, basert på funksjonane frå oppgåve 5. Implementer følgande to funksjonar:

```
void sub(R *r, R *x, R *y);
void div(R *r, R *x, R *y);
```

Funksjonen **sub** skal kalle **add**-funksjonen med motsatt forteikn på **y**. Gjer dette på ein slik måte at du ikkje endrar verdien til **y** som ein sideverknad av å kalle **sub**.

Funksjonen **div** skal kalle **mul**-funksjonen med den inverse verdien til **y**. (Dvs. at teljar og nemnar i **y** skal bytte plass.) Sjå også her til at du ikkje endrar verdien til **y** som ein sideverknad av å kalle **div**.

Oppgåve 7 (25%)

Du vil representere ein sekvens av rasjonelle tal som ei lenka liste. Du har definert følgande datatype for å representere ein node i ei slik liste:

```
struct node {
    R verdi;
    struct node *neste;
};
typedef struct node node_t;
```

Implementer følgande fire funksjonar for å manipulere slike lister:

```
int lengde(node_t *liste);
void sum(R *r, node_t *liste);
void gjennomsnitt(R *r, node_t *liste);
void minimum(R *r, node_t *liste);
```

Funksjonen **lengde** skal returnere lengden til ei lenka liste, altså talet på noder i lista. Du kan anta at lista er terminert med ein NULL-peikar.

Funksjonen **sum** skal rekne ut summen av dei rasjonelle tala som ligg i ei liste. Du skal skrive resultatet til det talet som **r** peiker på.

Funksjonen **gjennomsnitt** skal rekne ut gjennomsnittsverdien av dei rasjonelle tala som ligg i ei liste. Du skal skrive resultatet til det talet som **r** peiker på.

Funksjonen **minimum** skal finne det minste rasjonelle talet i ei liste og skrive resultatet til det talet som **r** peiker på.

For tomme lister skal **lengde** returnere 0, og dei andre funksjonane skal returnere utan å ha nokon verknad, dvs. utan å endra verdien som **r** peiker på.

For å implementere **mimimum** må du kunne samanlikne brøkar. Merk at dersom $\frac{a}{b} < \frac{c}{d}$ så vet vi også at $a * d < b * c$.