FAKULTET FOR TEKNOLOGI OG REALFAG

E K S A M E N

Emnekode: **MAS234**
Emnenavn: **Innebygde datasystemer for mekatronikk**

Dato: 12. desember 2019
Varighet: 4 timer

Antall sider inkl. forside: 9

Tillatte hjelpemidler: Alle trykte og håndskrevne hjelpemidler tillatt. Kalkulator tillatt.

Merknader:

- Kandidaten må selv kontrollere at oppgavesettet er fullstendig.
- Les nøye igjennom oppgavene slik at du forstår hva det spørres etter.
- Beskriv eventuelle antagelser du må gjøre dersom oppgaven er formulert uklart. Lesing av spesifikasjoner, herunder oppgavetekster, er en del av det dere blir testet i.
- All kildekode i oppgaveteksten er gitt i C++11. Der det spørres om implementasjon i en eller annen form, skal det programmeres C++ på papir.
- Det kan antas at header <iostream> er inkludert fra før, alle andre nødvendige includes må spesifiseres i løsningen.
- Kodesnuttene i oppgaveteksten er i en del tilfeller listet "frittstående". Det skal antas at disse kjøres i kontekst av en funksjon (eksempelvis inni en main-funksjon).

Oppgave 1 (10%)

Programforståelse.

Gitt følgende kodesnutt:

```
unsigned int x = 50;
unsigned int y = 100;

int a = x + y;

int* z = &a;
*z += 1;

std::cout << "a = " << a << ", z = " << *z << std::endl;
```

a) Kodesnutten er plassert inni en main-funksjon i en cpp-fil. Hva skrives ut når programmet bygges og kjøres?

a = 151, z = 151

b) Hvilken funksjon har & slik operatoren benyttes i kodesnutten?

Address-of (hente adressen til variabelen)

c) Kan variabelen a i kodesnutten deklareres const? Hvorfor/hvorfor ikke?

Nei, ikke når vi endrer den via peker. Vi kan heller ikke ha den const dersom vi IKKE endrer verdien via pekeren, det er nok at vi KAN gjøre det via pekeren. Dersom vi skal ha variabelen a const, så må pekeren z også være en peker til en const int.

d) Hva er forskjellen på pre-inkrementering (eksempelvis ++i) og post-inkrementering (eksempelvis i++)? Gi et kort eksempel på bruk hvor oppførselen er forskjellig.

Pre-inkrementering endrer verdien før den eventuelt leses/brukes. Post-inkrementering endrer den etter den eventuelt leses/brukes.

Eksempel; her vil a tilordnes verdien 0, mens b vil tilordnes verdien 1. Samtidig vil både i og j ha verdien 1 etter siste statement:

```
int i = 0;
int j = 0;
int a = i++;
int b = ++j;
```

Oppgave 2 (9%)

Funksjoner og løkker.

Det skal lages en funksjon som tar inn en array av double-elementer. Funksjonen skal returnere det største elementet.

a) Skriv funksjonsdeklarasjonen.

```
double largestElement(double* numbers, int length);
```

b) Lag implementasjonen til funksjonen fra deloppgave a).

```
double largestElement(double* numbers, int length)
{
    double largest = numbers[0]; // Assume first element is largest.

    for (int i = 1; i < length; ++i)
    {
        if (numbers[i] > largest)
        {
            largest = numbers[i];
        }
    }

    return largest;
}
```

c) Hva vil være viktig her dersom funksjonen skal være effektiv på store arrays? Eksempelvis arrays med 100.000 elementer eller mer.

Her er det viktig at vi faktisk sender inn en peker til array-en med elementene, og ikke en kopi. Når det gjelder POD-arrays av denne typen, så er det uansett pekere som overføres, vi kan ikke sende en kopi av array-en uten std::copy-kall etc..

Hadde vi derimot benyttet en std::vector eller lignende, i stedet for en array, så ville en referanse vært å foretrekke.

Oppgave 3 (8%)

Standardbibliotek.

Det skal lages en funksjon som tar inn en `std::vector` av `double`-elementer. Funksjonen skal fjerne det siste elementet (merk at endringen også skal være synlig på utsiden av funksjonen!).

a) Skriv funksjonsdeklarasjonen.

```
void removeLast(std::vector<double>& v);
```

b) Lag implementasjonen til funksjonen fra deloppgave a).

```
void removeLast(std::vector<double>& v)
{
    if (!v.empty())
    {
        v.pop_back();
    }
}
```

Oppgave 4 (10%)

Programforståelse.

Gitt følgende:

```
class Something
{
    public:
        int x;
    private:
        int y;
};

int main()
{
    Something s;
    s.x = 314;
    s.y = 157;
    return 0;
}
```

a) Hvorfor kompilerer ikke programmet?

Det forsøkes å lese en variabel som er deklarert private i klassen Something fra utsiden av klassen.

b) Nå skal klassen `Something` modifiseres slik at begge medlemsvariablene er deklarert private.

I tillegg skal det lages en funksjon som gir medlemsvariabelen `x` en verdi, og en funksjon som kan hente ut verdien til `x`. Kall funksjonene henholdsvis `setX` og `getX`.

Lag en ny deklarasjon til klassen som inkluderer endringene.

```
class Something
{
public:
    void setX(int x);
    void setY(int y);
private:
    int x_;
    int y_;
};
```

c) Lag implementasjonen til `getX` og `setX`. Dette skal lages på samme måten som hvis det hadde vært plassert i en egen implementasjonsfil.

```
void Something::setX(int x)
{
    x_ = x;
}

void Something::setY(int y)
{
    y_ = y;
}
```

d) Gi et eksempel på en vanlig konvensjon for filendelser på henholdsvis implementasjonsfiler og header-filer i C++.

.cpp og .h (flere andre varianter godtas også).

Oppgave 5 (15%)

Objektorientering.

Gitt følgende C++-klasser og main-funksjon.

En typisk bruk av filtrene vil være at vi har en array eller en std::vector hvor vi ønsker å kalle filter-funksjonen på hvert enkelt element i rekkefølge.

En annen bruk vil kunne være at vi leser en sensor med fast rate, f.eks. ti ganger i sekunder, og filtrerer verdiene vi leser inn før de brukes videre i programmet.

```
class Filter
{
public:
    virtual ~Filter() = default;
    virtual double filter(double value) = 0;
};



class NoPassFilter : public Filter
{
public:

    virtual ~NoPassFilter() = default;

    virtual double filter(double value) override
    {
        return 0.0;
    }

};


class PassThroughFilter : public Filter
{
public:

    virtual ~PassThroughFilter() = default;

    virtual double filter(double value) override
    {
        return value;
    }

};
```

```
// Oppgave 5 fortsettelse:


int main()
{

    Filter* filter = new NoPassFilter();

    const double unfilteredValue = 0.9;
    const double filteredValue = filter->filter(unfilteredValue);

    std::cout << "Unfiltered value is: " << unfilteredValue
              << ", filtered value is: " << filteredValue
              << std::endl;

    return 0;
}
```

a) Hva skrives ut dersom programmet bygges og kjøres?

```
Unfiltered value is: 0.9, filtered value is: 0
```

b) Det skal byttes filter. Vi ønsker å benytte et PassThroughFilter i stedet for filteret som er benyttet nå. Vis endrede linjer.

```
Filter* filter = new PassThroughFilter();
```

c) Lag en ny filterklasse som også arver fra Filter. Din nye filterklasse skal sende signalet rett igjennom («pass through») frem til første verdi lik 0.0 dukker opp, og fra og med dette tidspunktet skal filteret kun returnere 0.0.

Gi filterklassen et passende navn. Deklarasjon og implementasjon kan lages på samme sted, som i eksemplene over.

Flere løsninger godtas. Deklarasjon og implementasjon vises her sammen (ref. oppgaveteksten) -- i praksis vil det gjerne være ønskelig å skille dette i separate filer.

```
class ZeroPassTriggeredFilter : public Filter
{
public:

    ZeroPassTriggeredFilter()
        : active_(false)
    {}

    virtual ~ZeroPassTriggeredFilter() = default;

    virtual double filter(double value) override
    {
        // Eller gjerne en betingelse som sjekker et lite område
        // rundt 0.0, i og med at eksakt 0.0 kan være vanskelig å
        // oppnå dersom verdien er samplet fra en reell sensor.
        if (value == 0.0)
        {
            // Husk at vi har funnet verdien 0.0.
            active_ = true;
        }

        // Returnerer 0.0 hvis vi nå eller tidligere har funnet
        // verdien 0.0 i inn-signalet.
        return active_ ? 0.0 : value;
    }

private:

    bool active_;

};
```

Oppgave 6 (10%)

Systemmodellering.

a) I SysML benyttes blant annet aktivitetsdiagram for å modellere systemers oppførsel. Hvilke diagramtyper er egnet for å modellere systemers struktur?

Minst to av de følgende gir full poengsum her:
Block definition diagram
Internal block diagram
Package diagram

b) Hva kjennetegner en god systemmodell i MBSE-sammenheng?

Flere svar aksepteres; se forelesningsfoiler MBSE.

Oppgave 7 (10%)

Begrepsforståelse.

a) Hva er et namespace i C++ ?

Et namespace (navnerom) gir muligheten til å organisere entiteter (eksempelvis variabler, funksjoner og datatyper inkl. klasser) i et hierarki, slik at samme navn kan benyttes flere steder i ulike namespace. Ved korrekt bruk av namespace reduseres risikoen for feil som følge av navnekollisjoner, og det blir også tydelig hvor vi henter en gitt funksjon eller datatype fra (eks. `std::vector`).

Standardbiblioteket i C++ benytter eksempelvis namespace std, slik at vi kan benytte funksjonen max i navnerommet std ved å kalle `std::max(a, b)`.

b) Lag et namespace med navn controller, og opprett en const-variabel av typen int inni namespace-et. Gi variabelen et valgfritt navn og verdi 2.

```
namespace controller
{
    const int numberOfWheels = 2;
}
```

c) Hva gjør preprosessoren når vi bygger et C++-program?

Preprosessoren kjøres før kompilering. Den håndterer blant annet ut include-filer og preprosessor-direktiver slik som #defines, #ifndef etc. Resultatet etter preprosessor-steget er en enkelt fil klar til kompilering. Preprosessering er et av stegene når C++-programmet bygges.

d) Hva gjør kompilatoren når vi bygger et C++-program?

Kompilerer filen fra preprosseserings-steget til object-fil (maskinkode + informasjon til linkesteget). Dette kan inkludere et mellomsteg hvor assembly-kode for plattformen først genereres. Men det er ikke påkrevd å gjøre dette basert på C++-standarden.

Oppgave 8 (9%)

Kommunikasjon

a) Hva menes med begrepet integritet i sammenhengen overføring av digitale data?

Integritet i sammenhengen dataintegritet, betyr at vi kan være sikre (nok) på at data ikke er endret på vei fra avsender til mottaker.

b) Nevn to metoder som kan benyttes for å verifisere integritet på en overført digital melding. Hvilke typer feil kan detekteres?

CRC (Cyclic Redundancy Check). CRC er en feildetekterende kode med stor utbredelse. CRC-er er relativt gode til å ta burst-feil, altså feil hvor flere påfølgende bit er endret.

Paritetsbit. Dette er den enkleste formen for feildetekterende kode, og kan kun detektere et odde antall feil. Hvis det oppstår endringer i et partall antall bit, så vil dette ikke gi utslag i paritetsbit-et, og feilen vil dermed ikke detekteres. Paritetsbit er egnet til å detektere at noe er galt med overføringen for relativt korte meldinger hvor konsekvensene ikke er store ved feil som ikke oppdages. Metoden er svært utbredt, pga. enkel implementasjon og enkle beregninger (som ikke tar mye kjøretid).

c) Nevn en metode som kan benyttes for å korrigere for feil i en overført melding. Hvilke typer feil kan korrigeres med denne metoden?

Flere mulige svar her. Eksempelvis:

Raptor-code (eller mer generelt fountain-code). Benyttes blant annet i DVB-T. Kan korrigere for ønsket mengde feil ved å sende en bestemt mengde ekstra informasjon. Relativt robust mot både burst-feil og enkeltbit-feil. Relativt komplisert implementasjonsmessig, og også krevende kjøretidsmessig.

Oppgave 9 (6%)

Nettverk.

Det skal overføres data mellom 3-5 enheter. To av enhetene er henholdsvis en motorstyringsnode og en node med hastighetsregulator.

Fysisk avstand mellom nodene er 200-3000 meter.

a) Velg en passende type nettverk/kommunikasjonsstandard og begrunn valget.

En mulig løsning er å holde nodene for motorstyring og hastighetsregulator så nær hverandre som mulig, slik at eksempelvis RS422 kan benyttes (3000 meter er her for stor avstand, da RS422 på «normal» lav rate går 1500 meter). Det kan settes på en repeater på midten for å oppnå 3 km rekkevidde med RS422 uten å gå uvanlig lavt på datarate.

Eventuelt kan det velges en trådløs kommunikasjonsstandard (som LoRa), noe som i de fleste tilfeller vil være rimeligere enn kablet forbindelse. Men for motorstyringsdelen i lukket sløyfe vil en kablet løsning antageligvis være å foretrekke. Hvis mulig, så er det antageligvis best å anvende en fiberoptisk forbindelse f.eks. med en passende tranceiver for større distanser.

CAN-bus på vanlig twisted pair kan også være et alternativ, distanser på opp mot 5-6 km bør gå greit med høgkvalitetskabel så lenge hastigheten er satt ned til ca. 10 kbit/s. For avstander større enn 50 meter må følgende være oppfylt:

$$\text{Signaling Rate (Mbps)} \times \text{Bus Length (m)} \leq 50$$

Noe som ved en bus-lengde på 3000 meter gir at vi må under ca. 16 kbit/s hastighet for å overholde bit-timing på bus-en. (0,016 Mbps). Denne grensen er gitt av lyshastigheten. I praksis velger vi da 10 kbit/s for å få litt sikkerhetsmargin.

b) Hvilken topologi vil du benytte for oppkoblingen? Lag en enkel figur som illustrerer valget.

Flere mulige svar basert på valgene i a).

Oppgave 10 (7%)

Operativsystemer.

Forklar kort:

1) Hva er den viktigste forskjellen på en tråd og en prosess?

   En prosess har eget minne, en tråd deler minne med andre tråder i prosessen den tilhører.

2) Hva er prosessisolasjon (eng. «process isolation»)?

   Prosessisolasjon er en maskinvare- og operativsystemfunksjonalitet som forhindrer prosesser i å, ved en feil, lese og/eller endre minne som ikke tilhører prosessen. Prosessisolasjon er sterkt ønskelig, da det er en god mekanisme for å isolere feil til prosessen de eventuelt oppstår i. En lang rekke moderne operativsystem har prosessisolasjon.

   https://en.wikipedia.org/wiki/Process_isolation

Oppgave 11 (6%)

Single board computer vs. mikrokontroller. Operativsystem/ikke-operativsystem.

Velg typen embedded-maskin du mener er egnet for følgende to applikasjoner. Gi en kort begrunnelse for valget.

a) Styring av signalgenerator/funksjonsgenerator (faktisk generere signalene). Det er behov for sinus-signaler med frekvens opp til 20 kHz.

Her kan det være en grei løsning å anvende en mikrokontroller, da tidskravene er svært strenge, og det kan være enklere å oppnå nødvendig tidsoppløsning. Alternativt kan et spesialisert sanntidsoperativsystem eller en SBC uten operativsystem anvendes. Spesialiserte prosessorer av typen DSP kan også være en god løsning her for å syntetisere signaler (men det er neppe strengt nødvendig med slikt på 20kHz frekvens).

b) Logging av tidsserie-data fra solcellepanel-system (spenning, strøm, effekt, temperatur etc.), og tilgjengeliggjøring av loggede data på HMI (brukergrensesnitt) og skytjeneste på internett.

Her er det en fordel med innebygget støtte for skjerm, ethernet, nødvendige bus-systemer for å koble seg mot solcellepanel-kontrollerene etc.. Et standard operativsystem vil være et godt valg, det ser ikke ut til å være nødvendig med strenge tidskrav og sanntidsegenskaper.

UNIVERSITETET I AGDER

FAKULTET FOR TEKNOLOGI OG REALFAG

E K S A M E N

Emnekode: **MAS234**
Emnenavn: **Innebygde datasystemer for mekatronikk**

Dato: 11. desember 2018
Varighet: 4 timer

Antall sider inkl. forside: 10

Tillatte hjelpemidler: Alle trykte og håndskrevne hjelpemidler tillatt. Kalkulator tillatt.

Merknader:

- Kandidaten må selv kontrollere at oppgavesettet er fullstendig.
- Les nøye igjennom oppgavene slik at du forstår hva det spørres etter.
- Beskriv eventuelle antagelser du må gjøre dersom oppgaven er formulert uklart. Lesing av spesifikasjoner, herunder oppgavetekster, er en del av det dere blir testet i.
- All kildekode i oppgaveteksten er gitt i C++11. Der det spørres om implementasjon i en eller annen form, skal det programmeres C++ på papir.
- Det kan antas at header <iostream> er inkludert fra før, alle andre nødvendige includes må spesifiseres i løsningen.
- Kodesnuttene i oppgaveteksten er i en del tilfeller listet "frittstående". Det skal antas at disse kjøres i kontekst av en funksjon (eksempelvis inni en main-funksjon).

Oppgave 1 (7%)

Programforståelse.

Gitt følgende kodesnutt:

```
const int x = 3;
const int y = 2;
int z = 82;

const int a = x/y;
int* b = &z;

std::cout << "a = " << a << ", b = " << *b << std::endl;
```

a) Kodesnutten er plassert inni en main-funksjon i en cpp-fil. Hvis vi antar at cpp-filen kompileres og programmet kjøres: Hva skrives ut?

a = 1, b = 82

b) Hva betyr * på linje 6 i kodesnutten?

Det betyr at vi deklarerer en peker. I dette tilfellet en peker til en `int`.

c) Hvilken funksjon har * slik operatoren benyttes på siste linje i kodesnutten?

* er her dereference-operator, og anvendes på en pekervariabel for å hente ut verdien på pekeradressen.

d) Kodesnutten er plassert inni en main-funksjon og er opprinnelig laget for bruk på en PC med MS Windows 10. Du bestemmer deg for å skrive om programmet for bruk med Arduino. Hvor plasserer du kodesnutten nå, hvis den fremdeles skal kjøres én gang? Og må noe annet endres?

Det er ikke nødvendig å skrive om programmet for Arduino for å løse denne deloppgaven.

Koden kan eksempelvis plasseres i setup-funksjonen.

Det må gjøres endringer med tanke på å håndtere utskrift til terminal. Dersom serieport skal anvendes på Arduino-en, så kan det enten benyttes innebygget bibliotek med printf, eller det kan anvendes en kompakt versjon av iostream skrevet for Arduino.

Oppgave 2 (7%)

Funksjoner.

a) Skriv deklarasjonen til en funksjon som tar inn to heltall av valgfri heltallstype og returnerer en bool. Funksjonen skal hete isGreater.

```
bool isGreater(uint8_t first, uint8_t second);
```

b) Lag implementasjonen til funksjonen fra deloppgave a). Funksjonen skal returnere true hvis første argument er større enn andre argument, false ellers.

```
bool isGreater(uint8_t first, uint8_t second)
{
    return first > second;
}
```

Oppgave 3 (15%)

Løkker.

Følgende kodesnutt printer tall til standard output (vanligvis terminalvinduet).

```
const uint8_t n = 25;

for (uint8_t i = 0; i <= n; i++)
{
    std::cout << +i << std::endl;
}
```

a) Hva er første og siste tall som skrives ut?

Henholdsvis 0 og 25.

b) Hvor mange tall skrives ut?

26

c) Hvorfor bør variabelen ”n” være const i denne kodesnutten?

Fordi det ikke er behov for å endre på verdien etter initialisering.

d) Skriv en tilsvarende implementasjon som skriver ut hvert fjerde tall, fra og med 4, til og med 28.

```
for (uint8_t i = 4; i <= 28; i+=4)
{
    std::cout << +i << std::endl;
}
```

Oppgave 4 (8%)

Programforståelse.

Følgende swap-funksjon er skrevet for å bytte verdi mellom to variabler. Main-funksjonen er et eksempel på bruk av funksjonen.

```
void swap(int first, int second)
{
    const int temp = first;
    first = second;
    second = temp;
}

int main()
{
    int x = 314;
    int y = -2300;

    swap(x, y);

    std::cout << "x=" << x << " y=" << y << std::endl;
    return 0;
}
```

a) Vil programmet kompilere, og er det kjørbart? Hva printes ut hvis det kan kjøres?

Programmet lar seg kompilere, og er kjørbart. Utskrift til terminal:

x=314 y=-2300

b) Fungerer programmet i henhold til spesifikasjonen (altså bytter det verdien på variablene)? Hvis ikke, foreslå en endring ved å skrive en ny og fungerende swap-funksjon.

Nei, det bytter ikke om på verdiene. En mulig endring er å anvende referanser. Bruk av pekere på rett måte kan også være en god løsning her.

```
void swap(int& first, int& second)
{
    const int temp = first;
    first = second;
    second = temp;
}
```

c) Bør variablene x og y i main deklareres const? Hvorfor / hvorfor ikke?

Nei, bytte av verdi krever at vi skriver til variablene.

Oppgave 5 – Objektorientering (15%)

Gitt følgende C++-klasser:

```
class SensorData
{
public:
    SensorData(long t, double v)
        : timeStamp_us(t)
        , value(v)
    {
    }

    long timeStamp_us;
    double value;
};



class ISensorDataReceiver
{
public:
    virtual ~ISensorDataReceiver() { }
    virtual void setSensorData(SensorData& sd) = 0;
};



class TemperatureReceiver : public ISensorDataReceiver
{
public:
    TemperatureReceiver() { }

    virtual ~TemperatureReceiver() { }

    virtual void setSensorData(SensorData& sd)
    {
        if (timeStamp_us_ > sd.timeStamp_us)
        {
            timeStamp_us_ = sd.timeStamp_us;
            temperature_ = sd.value;
        }
    }

private:
    long timeStamp_us_;
    double temperature_;
};
```

Oppgave 5 - fortsettelse fra forrige side:

a) Skriv implementasjonen til en main-funksjon som:

I. Først oppretter et SensorData-objekt med tidsstempel 12345 og måleverdi 38.9

II. .. deretter oppretter et TemperatureReceiver-objekt,

III. .. og til slutt sender inn det opprettede SensorData-objektet til TemperatureReciver-instansen ved hjelp av setSensorData-funksjonen.

```
int main()
{
    SensorData sd(12345l, 38.9); // I

    TemperatureReceiver tr;      // II

    tr.setSensorData(sd);        // III

    return 0;
}
```

b) Hvorfor kan det ikke opprettes en instans av ISensorDataReceiver-klassen, og hva kaller vi denne typen klasser i C++ ?

(ISensorDataReceiver er et interface, og dette lages i C++11 ved bruk av typen klasse det spørres om).

Klassen ISensorDataReceiver har en pure virtual medlemsfunksjon (=0), og er dermed en abstrakt klasse. Abstrakte klasser kan ikke instansieres.

c) Formålet med sjekken i setSensorData-funksjonen er å sikre at vi kun tar i mot målinger som er nyere enn den forrige vi mottok.

Det har sneket seg inn en liten feil i setSensorData-funksjonen. Hva er feilen?

Det kreves at nytt tidsstempel er mindre enn eksisterende tidsstempel, noe som neppe gir mening. Det korrekte ville vært å kreve at tidsstempel på ny måling er større enn tidsstempelet på forrige måling:

```
if (sd.timeStamp_us > timeStamp_us_)
```

Oppgave 6 (8%)

Udefinert oppførsel.

a) Hva er udefinert oppførsel, slik det er definert i C++-standarden?

Udefinert oppførsel opptrer når gitte språklige regler i C++ ikke overholdes. Konsekvensen er at programmet kan gjøre «hva som helst», og vi har dermed et meningsløst og potensielt farlig program

b) Hvorfor er det viktig å unngå å skrive programmer hvor udefinert oppførsel forekommer?

Udefinert oppførsel kan føre til at programmet krasjer, påvirker andre programmer eller operativsystemet, endrer data i feil del av programmet etc. Det kan være vanskelig å detektere denne typen feil under testing, og det kan også gjøre programmet mindre porterbart da det ikke følger C++-standarden. Feil kan komme til overflaten når tilsynelatende urelaterte endringer innføres.

c) Gi et eksempel på udefinert oppførsel i C++.

Flere mulige svar. F.eks. integer overflow (signed integer), bruk av peker til objekt som er deleted, aksessere element etter siste eksisterende element i en array etc.

Oppgave 7 (5%)

Datatypen struct.

a) Lag en struct som inneholder feltene sensorId, timeOfValidity og measuredTemperature. Velg passende datatyper, og begrunn valgene kort.

Struct-en skal kun deklareres.

```
struct TemperatureMeasurement
{
    uint8_t sensorId;
    long timeOfValidity;
    float measuredTemperature;
};
```

b) Hva skiller en C++-struct fra en C++-klasse?

Standard synlighet på medlemsvariable og medlemsfunksjoner er public i en struct og private for en class (§ 11.2 i C++11-standarden).

Oppgave 8 (10%)

Begrepsforståelse.

a) Beskriv kort hva som skjer når vi kompilerer et C++-program.

Menneskelig lesbar C++-kode blir omformet assembly-kode. Dette skjer etter preprosessering, men før assembler og linking til kjørbar fil (hex-fil, elf-fil, a.out etc..). Det gis halv score på å omtale hele prosessen som kompilering.

(Det teller positivt med en figur som viser stegene med preprosessering, kompilering, assembler og linking men det er ikke et krav om figur i besvarelsen for full uttelling.)

b) Hva skiller "ordinær" kompilering fra krysskompilering?

Ved krysskompilering kompileres det for enn annen plattform enn det host-maskinen (maskinen som kompilerer) kjører på. Eksempelvis kryskompilering til en 8 bit Atmega128 på en Windows 10-maskin med x86-64-prosessor.

c) Under mikrokontrollerlab-en med AVR mikrokontrollere, ble det benyttet en Atmel ICE for skriving til flash-minnet på mikrokontrolleren (med mer).

Skjer skriving av flash-minnet på mikrokontrolleren før eller etter krysskompilering?

Etter.

Oppgave 9 (10%)

Mikrokontrollerkretser

a) Hvilken funksjon har typisk en spenningsregulator i en mikrokontrollerkrets?

Den skal forsyne kretsen med stabil og mest mulig støyfri spenning på et gitt nivå.

b) Nevn minst to typer spenningsregulatorer, og forklar hva som skiller disse.

Linear (f.eks. LDO) og switched mode (f.eks. buck).

En lineær spenningsregulator gir typisk en stabil og fin utspenning med lite støy. Imidlertid medfører den stort effekttap i form av varme dersom inn-spenningen er mye større enn ut-spenningen og strømmen er stor. Spenningsfallet blir direkte et tap som funksjon av strømmen.

En switched-mode strømforsyning (DC/DC) av typen buck er en step-down-konverter, og kan på samme måte som en Lineær spenningsregulator kun regulere en innspenning ned til et lavere nivå. Buck-konverteren er basert på at en spole og en kondensator tilføres energi når en transistor slås på, og deretter lades ut når transistoren er av (f.eks. gjennom en diode eller en ekstra transistor). Dette gjøres høgfrekvent, slik at lasten kan tilføres en lavere spenning med rimelig lite rippel. Imidlertid støyer switched mode strømforsyninger generelt mer enn en lineær strømforsyning. Den store fordelen til en switched mode spenningsregulator er at den i de fleste tilfeller vil ha et langt lavere tap enn en lineær spenningsregulator når innspenningen er høg sammenlignet med utspenningen.

c) En rød LED med spenningsdropp $V_f = 2.3\ V$ skal styres av en mikrokontroller med supplyspenning $V_s = 5\ V$. Tegn et enkelt kretsskjema for hvordan dette bør kobles. Skriv på komponentverdier, og vis hvordan du kommer frem til disse.

Det er ønskelig med maksimal lysstyrke, i og med at anvendelsen er utendørs. Komponentenes levetid er imidlertid også viktig.

<u>Se Vedlegg A for utsnitt fra LED-ens datablad</u>.

Tegningen skal inneholde motstand i serie med LED i rett retning sammenlignet med supplyspenningen, og koblet til en pinne på mikrokontrolleren. Det bør også tas en

vurdering på om pinnen på mikrokontrolleren kan source eller sinke nødvendig strøm.

Verdi på motstanden bør være: $R_{min} = \frac{V_s - V_f}{I_{max}} = 90\ \Omega$.

c) Hvis mikrokontrolleren i deloppgave c) er av AVR-type; hvilke steg er nødvendige for å kunne styre den programmatisk når du har koblet opp i henhold til kretsskjemaet ditt fra c) ? (Hva må du gjøre i f.eks. et C++-program for å styre LED-en)?

Rett pinne på rett port må settes som utgang. Deretter på det skrives lav til pinnen for å få lys ved sinking eller høg ved sourcing.

Oppgave 10 (5%)

Nettverkstopologier.

Hvilke typer (én eller flere) nettverkstopologier er støttet med CAN? Tegn ett eller flere eksempler.

Bus (point-to-point kan også aksepteres, men kun i tillegg til bus).

Oppgave 11 (5%)

Operativsystemer I.

Forklar kort:

1) Hva de viktigste oppgavene til et operativsystem er,

   Ikke-prioritert rekkefølge:
   - Ressurshåndtering; fordeling og tilgangsstyring. Scheduling av prosesser, håndtering av systemminne og styre og aksessere IO-ressurser.
   - Tilrettelegge for videreutvikling. Eksempelvis ved å håndtere hardware-abstraksjon, tilby tjenester, være vedlikeholdbare (system for patching og oppgradering av OS og gjerne også applikasjoner).
   - Bekvemmelighet. Gi applikasjonsprogrammereren et i stor grad programmeringsspråk- og hardware-uavhengig ABI og API. ABI utnyttes for å kunne la programmer skrevet i ulike språk samhandle (gjennom funksjonskall), og API-er lar programmereren utnytte (i stor grad) hardware-uavhengige høgnivå-grensesnitt for interaksjon med operativsystemet og derigjennom også maskinvareressurser. Eksempelvis bruk av innebygget støtte for nettverkskommunikasjon (ethernet, CAN etc.).

2) hvilke egenskaper i operativsystemet som er mest sentrale når det skal anvendes som kjøremiljø for styring av tidskritiske fysiske systemer. (Eksempelvis styring av et fysisk system med rask dynamikk).

   Sanntidsegenskaper. IO-støtte (tilgjengelige drivere for nødvendig hardware).

Oppgave 12 (5%)

Operativsystemer II.

Foreslå et passende operativsystem for hvert av de følgende tilfellene. Gi en 1-2 setningers begrunnelse for hvert svar.

a) Utvikling av desktop-applikasjon for registrering av måledata i et laboratorium.

   MS Windows, MacOS eller Ubuntu Linux. Det viktige her er å ha tilgjengelig gode rammeverk for datainnsamling og presentasjon (GUI, biblioteker for formatstøtte

etc.)

b) Lukket sløyfe-styring av en selvbalanserende robot. (Inkludert indre sløyfer.)

Her vil det være naturlig å velge et operativsystem med gode sanntidsegenskaper, eksempelvis Linux-variant med sanntids-patch, QNX, VxWorks eller lignende.

Vedlegg A – Utsnitt fra datablad

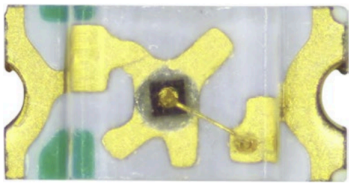
SMD LED

Red 2.1...2.6 V 0603

RND components

SPECIFICATION:

Package	0603
Light colour	Red
Transmission angle	130 °
Case colour	Transparent
Housing type	SMD
Conducting-state voltage max.	2.6 V
Conducting-state voltage min.	2.1 V
Dominant wavelength	631 nm
Forward current	30 mA
Forward voltage	2.1...2.6 V
Luminous intensity	80 mcd
Luminous intensity max.	80 mcd
Luminous intensity min.	80 mcd
Off-state current	10 µA
Off-state voltage	5 V
Operating temperature	-40...+85 °C
Peak wave length	645 nm
Power	78 mW
Technology	AlGaInP



UNIVERSITETET I AGDER

TEKNOLOGI OG REALFAG

Emnekode:
Emnenavn: Innebygde datasystemer for mekatronikk

Dato: 19. desember 2017
Varighet: 4 timer

Antall sider inkl. forside: 7

Tillatte hjelpemidler: Alle trykte og håndskrevne hjelpemidler tillatt. Kalkulator tillatt.

cfMerknader: Kandidaten må selv kontrollere at oppgavesettet er fullstendig.

- Les nøye igjennom oppgavene slik at du forstår hva det spørres etter.
- Beskriv eventuelle antagelser du må gjøre dersom oppgaven er formulert uklart. Lesing av spesifikasjoner, herunder oppgavetekster, er en del av det dere blir testet i.
- All kildekode i oppgaveteksten er gitt i C++11. Der det spørres om implementasjon i en eller annen form, skal det programmeres C++ på papir.
- Det kan antas at header <iostream> er inkludert fra før, alle andre nødvendige includes må spesifiseres i løsningen.
- Kodesnuttene i oppgaveteksten er i en del tilfeller listet "frittstående". Det skal antas at disse kjøres i kontekst av en funksjon (eksempelvis inni en main-funksjon).

Velg en byggeblokk.

Oppgave 1 (7%)

Programforståelse.

Gitt følgende kodesnutt:

```
const int x = 3;
const int y = 2;
int z = 42;

const int a = x*y;
int& b = z;
b++;

std::cout << "a = " << a << ", b = " << b << std::endl;
```

a) Kodesnutten er plassert inni en main-funksjon. Hva er spesielt med main-funksjonen i et C++-program?

Den globale funksjonen main er startpunktet i programmet. Dette gjelder i de tilfellene hvor programmet kjører på et operativsystem. For programmer som eksempelvis skal kjøres direkte på en mikrokontroller, er startpunktet definert av implementasjonen (f.eks. bestemt av kompilatorprodusenten). http://en.cppreference.com/w/cpp/language/main_function)

b) Hva skrives ut?

a = 6, b = 43

c) Dersom kodesnutten utvides med følgende linje på slutten. Hva skrives ut i tillegg?

std::cout << "z = " << z << std::endl;

z = 43

d) Hva betyr & etter int på linje 6?

Dette betyr at vi deklarerer en referanse. Variabelen b har følgelig type ”referanse til int” og refererer (er et alias) til variabelen z.

Oppgave 2 (7%)

Funksjoner.

a) Skriv deklarasjonen til en funksjon som tar inn en int og returnerer en bool. Funksjonen skal hete checkSubjectId.

```
bool checkSubjectId(int subjectId);
```

b) Lag implementasjonen til funksjonen fra deloppgave a). Funksjonen skal returnere true hvis tallet som sendes inn er 234, false ellers.

```
bool checkSubjectId(int subjectId)
{
    return subjectId == 234;
}
```

Oppgave 3 (15%)

Løkker.

Følgende kodesnutt printer tall til standard output (vanligvis terminalvinduet).

```
const uint8_t n = 25;

for (uint8_t i = 0; i < n; ++i)
{
    std::cout << +i << std::endl;
}
```

a) Hva er første og siste tall som skrives ut?

Henholdsvis 0 og 24.

b) Hvor mange tall skrives ut, og skrives de ut etter hverandre horisontalt eller vertikalt?

25 tall skrives ut. Tallene skrives ut etter hverandre vertikalt pga. Linjeskift (std::endl).

c) Kan variabelen ”i” være const i denne kodesnutten?

Nei, den kan ikke være const fordi variabelen i må endre verdi etter at den er opprettet. I dette tilfellet benyttes i som indeks i for-løkken, og ++i inkrementerer (øker) verdien på i med 1 for hver iterasjon (”runde”).

d) Skriv en tilsvarende implementasjon som skriver ut alle oddetall fra og med 5 til og med 131. Bruk den typen løkke som er best egnet for formålet.

Det bør benyttes en for-løkke her, men bruk av andre løkker kan aksepteres. En mulig løsning:

```
for (uint8_t i = 5; i <= 131; i+=2)
{
    std::cout << +i << std::endl;
}
```

En annen variant:

```
for (uint8_t i = 5; i <= 131; )
{
    std::cout << +i << std::endl;
    i = i + 2;
}
```

Oppgave 4 (8%)

Løkker.

Følgende kodesnutt er skrevet for å printe heltallene fra og med n til og med 0 i synkende rekkefølge til terminalen.

```
const uint8_t n = 10;

for (uint8_t i = n; i >= 0; --i)
{
    std::cout << +i << std::endl;
}
```

a) Er tallet 0 ett av tallene som printes til terminalen når programmet kjøres?
Ja.

b) Kodesnutten i denne oppgaven inneholder en rimelig ugrei feil. Hva er feilen, og hvordan kan denne rettes opp? (Flere ulike løsninger vil bli godtatt her).

Merk: Feilen gjelder ikke programsnutten i oppgave 3, selv om disse ligner noe strukturmessig.

Dette blir en evig løkke. Sjekken i >= 0 vil aldri bli false, da i er unsigned og følgelig ALLTID vil være >= 0. En mulig løsning er å benytte heltall med fortegn (bytt ut uint8_t med int8_t begge steder).

Oppgave 5 – Objektorientering (15%)

Gitt følgende C++-klasse:

```
class PidController : public IGenericController
{

public:
    PidController();
    virtual ~PidController();

    void setPid(double p, double i, double d);

private:

    void setP(double p);
    void setI(double i);
    void setD(double d);
};
```

a) Skriv implementasjonen til en funksjon som oppretter en PidController-instans og setter regulatorparametrene til: P=4.2, I=0.5, D=0.0.

```
int main()
{
    PidController pidController;
    pidController.setPid(4.2, 0.5, 0.0);

    return 0;
}
```

b) Kan PidController-instansen være const? Hvorfor / hvorfor ikke?

Nei, vi kan ikke deklarere variabelen pidController const. Medlemsfunksjonen setPid, som kalles senere, er ikke deklarert const, og denne kan følgelig modifisere eventuelle medlemsvariable (selv om det ikke er noen her nå).

c) PidController arver fra IGenericController. IGenericController har en destructor som er deklarert virtual. Hva betyr dette?

Bruk av virtual fører til at det opprettes en vtable for den gitte medlemsfunksjonen. Den får dynamisk binding. Hvis det opprettes en IGenericController-peker som peker til et PidController-objekt, så vil destructoren til både IGenericController OG PidController bli kalt når delete-operatoren benyttes på pekeren.

Uten bruk av virtual, blir kun foreldreklassens (IGenericController)-destructor kalt. Dette kan resultere i udefinert oppførsel dersom klassen har én eller flere andre virtual-deklarerte funksjoner.

http://en.cppreference.com/w/cpp/language/virtual

d) PidController arver fra IGenericController. IGenericController har en funksjon setTimeStepLength(double dt) som er deklarert pure virtual. Hva betyr dette for oss når vi lager klassen PidController?

Merk: Det er ikke listet kildekode for IGenericController, da denne ikke er nødvendig for å løse oppgaven.

Når medlemsfunksjonen setTimeStepLength er deklarert pure virtual, så er klassen den tilhører abstrakt. Ved arv fra en abstrakt parent-klasse har vi to valg: La child-klassen også være abstrakt, eller implementere alle pure virtual funksjoner.

I praksis benyttes pure virtual funksjoner ofte i grensesnitt for å ”tvinge” den som arver til å implementere bestemte funksjoner/metoder.

En abstrakt klasse er en klasse som ikke kan instansieres ref.
http://en.cppreference.com/w/cpp/language/abstract_class

Oppgave 6 (10%)

Programforståelse.

En kryptert beskjed er mottatt og må dekrypteres. Følgende program utfører dekrypteringen:

```
void fun(char* message, unsigned int startIndex, unsigned
int stopIndex)
{
    for (unsigned int i = startIndex; i > stopIndex;)
    {
        std::cout << message[--i];
    }
    std::cout << std::endl;
}


void decryptMessage(char* message)
{
    const unsigned int christmasEve = 24;

    std::cout << "Navnet er: " << std::endl;

    fun(message, christmasEve/2 - 2, 0);
}


int main()
{
    char secretMessage[] = "naihsadraknu-gnoj";

    decryptMessage(secretMessage);

    return 0;
}
```

a) Beskriv hva tredje parameter til fun-funksjonen gjør.
Tredje parameter til fun-funksjonen er stopIndex. Den bestemmer når (på hvilken index) for-løkken i funksjonen skal stoppe.

b) Basert på innholdet i funksjonen "fun", foreslå et bedre og mer beskrivende navn på funksjonen. Navnet skal være skrevet C++-teknisk sett korrekt, og gjøre det unødvendig med kommentarer i koden for å forklare hva funksjonen gjør.
printReverse

c) Hva er den hemmelige beskjeden? Hint: En mye omtalt Kim.

kardashian

Oppgave 7 (4%)

Lag en enumerator som inneholder de mulige verdiene red, green og blue. Gi enumeratoren et beskrivende navn. Det skal *ikke* opprettes en instans av enumeratoren, den skal kun defineres.

```
enum Color
{
    red,
    green,
    blue
};
```

Oppgave 8 (10%)

Begrepsforståelse.

a) Beskriv kort hva som skjer når vi kompilerer et C++-program.

Den tekstlige programkoden blir transformert / oversatt til et annet språk. Språket det oversettes til er typisk maskinkode (binær kode basert på instrukssjonssettet til datamaskinen vi planlegger å kjøre programmet på). Men det må ikke være maskinkode.

Bygging av et C++-program omfatter typisk tre steg; preprosessering, kompilering og linking. Dette omtales i en del sammenhenger som ”kompilering”, selv om det er noe upresist.

b) Hva skiller ”ordinær” kompilering fra krysskompilering?

Ved krysskompilering genereres det maskinkode (eller annen form for kode) ment for en annen plattform enn den kompileringen utføres på.

Ved ”ordinær” kompilering, genereres det maskinkode for plattformen kompileringen utføres på.

c) Hva mener vi med ”host” og ”target” når vi utvikler programvare for et innebygget datasystem? Tegn gjerne en enkel figur for å illustrere.

Host er maskinen utvikleren sitter på (f.eks. med et integrert utviklingsmiljø (IDE)).

Target er maskinen det utvikles programvare for.

Oppgave 9 (10%)

Mikrokontrollerkretser og CAN-bus

a) Forklar kort hva en avkoblingskondensator er, og hvordan disse normalt bør plasseres relativt til f.eks. en mikrokontroller.

Avkoblingskondensatoren er en kondensator som benyttes til å dekoble/avkoble en del av kretsen fra en annen. Først og fremst med tanke på å forhindre spredning av høgfrekvente, uønskede signaler (støy). Det benyttes ordinære kondensatorer som avkoblingskondensatorer, men typisk rimelig raske kondensatorer med forholdsvis lav kapasitans.

Avkoblingskondensatoren representerer en liten lokal energireserve der den er plassert. Når strøm eksempelvis trekkes i kortvarige pulser av en mikrokontroller, vil avkoblingskondensatoren fungere som en shunt mot jord (tett inntil mikrokontrolleren) og de skarpeste "kantene" fjernes fra signalet som brer seg videre utover i kretsen.

Avkoblingskondensatoren bør typisk plasseres fysisk nær kretselementet som skal avkobles.

b) Hva bør være foretrukket jordingsstrategi når vi designer et kretskort med flere relativt effektkrevende integrerte kretser (eller andre komponenter som har høyt strømtrekk)?

(Det samme gjelder også når vi designer en krets bestående av flere "break-out-boards" etc..).

Stjernejording. Dedikert jordplan på kretskortet.

c) Det er målt 100 kOhm motstand mellom Can High og Can Low på en CAN-bus. Hva kan dette indikere? Vil CAN-bus-en fungere i et slikt tilfelle?

Bussen er høgimpedant. Dette indikerer manglende termineringsmostand. Den fysiske CAN-bus-en skal ved bruk av kobberpar termineres som beskrevet her: http://www.ni.com/white-paper/9759/en/ (basert på ISO 11898).

Det er lite sannsynlig at CAN-bus-en vil fungere i dette tilfellet.

Oppgave 10 (7%)

Nettverkstopologier.

Tegn opp tre valgfrie vanlige nettverkstopologier og navngi disse.

F.eks. tre av disse: stjerne, bus, ring, mesh.

Oppgave 11 (7%)

Sanntidssystemer.

Velg de to punktene du mener beskriver et sanntidssystem best. (Skriv setningene fullt ut i besvarelsen).

- Et sanntidssystem må regne korrekt og ha svært god ytelse (kunne utføre mange beregninger raskt).
- Et sanntidssystem må forholde seg til tidsfrister.
- Et sanntidssystem må ha resultatet av beregningene klart til rett tid.
- Et sanntidssystem må har mange innganger og utganger.

UNIVERSITETET I AGDER

FAKULTET FOR TEKNOLOGI OG REALFAG

P R Ø V E E K S A M E N

Emnekode: **MAS234**
Emnenavn: **Innebygde datasystemer for mekatronikk**

Dato: MAS234 PRØVEEKSAMEN 2017
Varighet: Eksamen har 4 timers varighet, dette prøvesettet inneholder færre oppgaver enn en faktisk eksamen.

Antall sider inkl. forside: -

Tillatte hjelpemidler: Alle trykte og håndskrevne hjelpemidler tillatt. Kalkulator tillatt.

Merknader: Kandidaten må selv kontrollere at oppgavesettet er fullstendig. Les nøye igjennom oppgavene slik at du forstår hva det spørres etter. Beskriv eventuelle antagelser du må gjøre dersom oppgaven er formulert uklart. Lesing av spesifikasjoner, herunder oppgavetekster, er en del av det dere blir testet i. All kildekode i oppgaveteksten er gitt i C++11. Der det spørres om implementasjon, skal det programmeres på papir. Det kan antas at header <iostream> er inkludert fra før, alle andre nødvendige includes må spesifiseres i løsningen.

Oppgave 1

Gitt følgende kodesnutt:

```
const int x = 3;
const int y = 2;
int z = 42;

const int a = x*y;
int& b = z;
b++;

std::cout << "a = " << a << ", b = " << b << std::endl;
```

a) Hva skrives ut?

a = 6, b = 43

b) Kan z være const i denne programsnutten? Hvorfor / hvorfor ikke?

Nei, z kan ikke deklareres const. Hvis z er const, så må også referansen til z (altså variabelen b) være const, noe som gjør at vi ikke kan inkrementere b slik det gjøres i kodesnutten. (Inkrementering innebærer å skrive til variabelen.)

Oppgave 2

Begrepsforståelse.

a) Beskriv kort hva en ”toolchain” er i forbindelse med krysskompilering. Nevn to av de viktigste komponentene i en toolchain.

En toolchain for krysskompilering er en samling verktøy som gjør det mulig å lage kjørbare kode (”programmer”) for en maskin som har en annen arkitektur enn den platformen hvor krysskompilatoren kjører. Toolchain-en kan også inneholde f.eks. debuggingsverktøy, ferdigkompilerte bibliotek for target-plattform etc..

Et eksempel kan være krysskompilering og linking av programvare for en Rapsberry Pi 3 med ARM-prosessor på en host-maskin som kjører f.eks. Ubuntu Linux på en 64 bit intel x86-plattform.

Et annet eksempel kan være krysskompilering for en AVR mikrokontroller på f.eks. en host-maskin som kjører Windows eller MacOS på en intel x86-plattform.

De viktigste verktøyene i en generell toolchain er kompilator og linker, mens det ved krysskompilering blir mer korrekt å si krysskompilator og linker.

b) Vi har krysskompilert programmet vårt, og har fått ut en kjørbar fil. Vil denne kjørbare filen generelt sett kunne kjøres på host-maskinen? Hvorfor / hvorfor ikke?

Nei, i dette vil ikke være mulig i det generelle tilfellet. Når det er snakk om krysskompilering, så kompileres programmet for en target-plattform med et annet instruksjonssett enn det vi finner på host-plattformen. Binærkoden/maskinkoden vil dermed ikke vil kjøre korrekt (eller kjøre i det hele tatt) på host-plattformen (hvor vi utførte krysskompileringen).

Dette er riktignok en sannhet med modifikasjoner, da vi i mange tilfeller kan lage en virtuell maskin på host-maskinen (f.eks. ved hjelp av VirtualBox eller qemu), slik at vi faktisk kan kjøre programmer for f.eks. ARM-prosessorer på en maskin med ”tradisjonell” intel x86 eller AMD64 CPU. I dette tilfellet kjøres instruksjonene i det krysskompilerte programmet vårt på den virtuelle CPU-en, som oversetter dem til instruksjoner for den fysiske CPU-en på host-maskinen.

Oppgave 3

Arrays. Gitt følgende funksjon ”arrayFun”:

```
void arrayFun()
{
    int magNumbers[] = {1, 3, 7, 11, 24, 42};

    // a)
    std::cout << "a = " << magNumbers[2] << std::endl;

    // b)
    std::cout << "b = " << *magNumbers << std::endl;
}
```

a) Hva skrives ut av linjen etter kommentar a) ?
7 (med linjeskift etter hvis vi skal være pirkete..)

b) Hva skrives ut av linjen etter kommentar b) ?
1 (dette blir det samme som å skrive ut magNumbers[0]

Oppgave 4

Funksjoner.

a) Skriv deklarasjonen til en funksjon som tar inn to heltall av samme datatype og returnerer en bool. Funksjonen skal hete firstIsLargest. Velg datatype selv.

```
bool firstIsLargest(int first, int second);
```

b) Lag implementasjonen til funksjonen fra deloppgave a). Funksjonen skal returnere true hvis den første parameteren er størst, false hvis ikke.

```
bool firstIsLargest(int first, int second)
{
    return first > second;
}
```

Oppgave 5

Bruk av SSH for fjerntilgang til innebygget datasystem.

Ved hjelp av en SSH-klient kan vi logge oss på en datamaskin som kjører eksempelvis Dropbear eller OpenSSH-server over et IP-nettverk. Dropbear ble benyttet under obligatorisk lab i MAS234.

a) Hva bør sjekkes dersom følgende feilmelding dukker opp ved forsøk på tilkobling med SSH? Nevn de to-tre første sjekkpunktene dine i et slikt tilfelle.

   ssh: connect to host 10.0.0.3 port 22: Network is unreachable

   Her er det viktige å vise frem "feilsøkingsferdigheter". Det er ikke et låst sett med punkter som er korrekt, men noen gode punkter er:

   Sjekke at power på target er på og at operativsysemet har startet uten feil. Sjekke at nettverkskabel er koblet til. Sjekke at korrekte IP-adresser er satt (og at andre nettverksinnstillinger er OK på begge maskiner). Forsøke å pinge ip-adressen 10.0.0.3 fra host-maskinen. Sjekke at dropbear eller annen SSH-server er installert på target-maskinen, og at denne kjører og lytter på port 22.

b) Hvilke krav vil du sette til et passord for bruk til SSH-innlogging på et innebygget datasystem over internett? La oss si at maskinen styrer lastesystemet på et skip.

   Se gode råd her for tips: https://www.nsm.stat.no/blogg/passordrad/

   Og mer konkret og kortere "liste" her: https://www.stopthinkconnect.org/resources/preview/tip-sheet-passwords-and-securing-your-accounts

c) Hvis du ofte må logge inn på maskinen med SSH; hva kan gjøres for å få ned "arbeidsmengden" ved å taste et langt og sikkert passord – samtidig som graden av sikkerhet bevares eller økes? Svaret skal begrunnes.

   Bruk av public key authentication er brukervennlig samtidig som det kan gi høg grad av sikkerhet. Se generelle tips her: https://help.ubuntu.com/community/SSH/OpenSSH/Keys

   I praksis må man generere et public/private nøkkelpar (slik vi gjorde for bruk av

Github tidlig i øvingsopplegget), og plassere public-nøkkelen i authorized keys på maskinen som kjører ssh-server.

Oppgave 6

Løkker.

Følgende kodesnutt printer tall til standard out (vanligvis terminalvinduet).

```
const uint32_t n = 8;

for (uint8_t i = 5; i < n; ++i)
{
    std::cout << +i << std::endl;
}
```

a) Hva er første og siste tall som skrives ut?
   5 først 7 sist.
b) Hvor mange tall skrives ut, og skrives de ut etter hverandre eller under hverandre?
   3, under hverandre.
c) Hva gjør + foran i?
   + promoterer unsigned-variabelen til et tall som kan printes. Alternativet, som på mange måter er ”ryddigere”, er å benytte static_cast<int>(i)

Oppgave 7

Bruk av const.

a) Hva betyr det dersom vi deklarerer en variabel const?
Kompilatoren passer på at vi ikke skriver til variabelen etter at den er opprettet.

b) Gi et kort eksempel (maksimalt åtte linjer) med bruk av const.

Her benyttes const for å sikre at det ikke utilsiktet skrives til innparameteren r eller variabelen pi (pi må naturlig nok kun beregnes én gang, da det er en konstant).

```
#include <cmath>

const double pi = std::acos(-1);

void computeAreaOfCircle(const double r, double& area)
{
   area = pi * r * r;
}
```

c) Kompilerer det følgende programmet, hvis ikke – hvorfor?

```
#include <iostream>

int main()
{
    const double dt = 1.0e-6; // One microsecond sampling time
    const double acceleration_z = 9.81; // m/(s*s)

    int numberOfTimesteps = 100000;

    velocity = acceleration_z * dt * static_cast<double>(numberOfTimesteps);

    std::cout << "Velocity after " << numberOfTimesteps << " time steps";
    std::cout << " w/constant accl: " << velocity << " [m/s]" << std::endl;
}
```

Nei programmet kompilerer ikke, da variabelen velocity ikke er deklarert. C++ er et strengt typet språk, så alle variabler må deklareres eksplisitt med type.

Litt tilleggsinfo: I C++11 og nyere kunne vi benyttet ”auto” for å få typen automatisk dedusert/avledet fra initialiseringsverdien (altså det på høyre side av assignment i dette tilfellet). Auto har ikke vært dekket i forelesningene, og er følgelig ikke pensum.
http://en.cppreference.com/w/cpp/language/auto

d) Gi to eksempler hvor det med fordel kan benyttes const bør benytte const i et C++-program, og hva const gjør i disse tilfellene.

Tre eksempler:

- Når vi deklarerer variabler som det ikke senere skal skrives til -- som i oppgave 7 b).
- Når vi skal ta inn parametre til en funksjon – og ikke har til hensikt å skrive til disse. Dette er spesielt viktig når vi sender inn referanser, da utilsiktet skriving til en referanse vi får inn som en parameter til en funksjon vil ha effekt utenfor funksjonen. Kompilatoren vil nekte oss å sende inn en referanse til en const-variabel hvis parameteren ikke er deklarert const.
- Når vi deklarerer funksjoner tilhørende klasser kan disse deklareres const. Dette forteller kompilatoren at vi ikke har tilhensikt å skrive til medlemsvariablene i klassen, og all skriving til medlemsvariable vil dermed gi kompileringsfeil.

```
class ProportionalController
{

public:

    // const her betyr at vi "lover" å ikke skrive til
    // medlemsvariablene (f.eks. p_) når vi implementerer
    // getPValue()-funksjonen. Kompilatoren passer da på
    // at vi ikke gjør dette ved en feil.
    void getPValue() const;

private:

    double p_;
};
```

Oppgave 8

Structs.

a) Skriv deklarasjonen til en struct med navn "Fluid" som inneholder tre felt: id, pricePerLitre og litre. Velg selv datatype på feltene.

```
struct Fluid
{
    int id;
    float pricePerLitre; // Bør ha flyttall-type.
    float litre;         // Bør ha flyttall-type.
};
```

b) Lag en funksjon som tar inn en array med struct-typen du definerte i a) og summerer prisen for alle elementene. Funksjonen skal returnere prisen.

Hint: Prisen for ett element i arrayen er pricePerLitre ganget med antall liter.

Merk: Deloppgave b) kan løses selv om deloppgave a) ikke er besvart korrekt.

Det finnes (etter forelesers mening) ingen robust og "brukervennlig" fremgangsmåte for å finne antall elementer i en innebygget C eller C++-array. En grei rett-frem-løsning er å sende inn antall elementer som en ekstra parameter.

En bedre løsning vil i mange tilfeller være å benytte en "container-type". Container-typen er gjerne en klasse. Vi kan lage denne selv ved behov, men vi foretrekker typisk å benytte noe vi finner i standardbiblioteket. Eksempelvis std::vector.

Det følgende er en OK og kompakt løsning uten bruk av standardbibliotek eller klasser:

```
float computeTotalPrice(Fluid fluidElements[], int numberOfElementsInArray)
{
    float totalPrice = 0.0;

    for (int ii = 0; ii < numberOfElementsInArray; ++ii)
    {
        totalPrice += fluidElements[ii].litre * fluidElements[ii].pricePerLitre;
    }

    return totalPrice;
}
```

Oppgave 9

Mikrokontrollerkretser – Krystall (med utdrag fra datablad). Tegn kretsskjema for en AtMega128 mikrokontroller med strømforsyning og ekstern krystall.

Det er ikke nødvendig å tegne programmeringsgrensesnitt.

(Merk: jeg har ikke lagt ved datablad til prøveeksamen).

Lager ikke LF på denne. Krav til løsning er på nivå med krav til kretsskjema for strømforsyningskrets på lab. Husk å tegne på alle nødvendige signaler.

Oppgave 10

To programvareutviklere er leid inn for å lage styrings- og datainnsamlingsprogramvare for en solcellefarm. Prosjektet er beregnet å ta to måneder (kalenderuker) pluss testing.

a) Se for deg at du jobber i en hovedsakelig mekanisk orientert bedrift, og at det er din oppgave å argumentere overfor ledelsen for bruk av GIT versjonskontroll i dette prosjektet.

   Besvarelsen skal beskrive de viktigste fordelene ved bruk av versjonkontroll-programvare generelt. Se video fra forelesningen.

b) Hva har GIT til felles med Linux? (Ikke teknisk, men opphavsmessig).

   Linus Torvalds er skaperen av både operativsystemet Linux og det distribuerte versjonskontrollsystemet Git. Begge deler er lisensiert som fri programvare.

   (Linus Torvalds er ikke den eneste utvikleren, Linux videreutvikles nå vha. en ”community”-basert utviklingsmodell.)