Oppgåvesamling i EDB tilpassa digital signalbehandling.

1. Lag eit program for innlesing av tre tal x, y og z og for å finne og skrive ut produktet og summen av dei tre tala med høveleg tekst.
2. Lag eit program for innlesing av tre tal, utrekning og utskrift av gjennomsnittet av tala med høveleg tekst. Programmet skal gå i ei sløyfe med innlesing og utrekning til gjennomsnittet er 0.
3. Lag eit program for utrekning av eit "glidande middel" av innleste tal. Programmet skal gå i ei sløyfe med innlesing av eitt tal, utrekning og utskrift av gjennomsnittet av dei tre siste tala. Innlest verdi lik 999 skal stanse programmet.

   Framlegg til algoritme:
   Nullstill variablar.
   Gjenta
       Les inn eit tal x
       Rekn ut y som gjennomsnittet av x, xsist og xforsist
       Skriv ut y
       Flytt xsist inn i xforsist
       Flytt x inn i xsist
   Til x er 999

   Merknad:
   Dersom dei innleste tala er ein sekvens av målte spenningar, kan det "glidande middel" oppfattast som eit filtrert signal der raske endringar (høge frekvensar) er glatta ut - dvs. datamaskinen fungerer som eit lavpass-filter. I seinare oppgåver vil vi bruke desse notasjonene:

   y0 er det sist utrekna filtrerte signal (utsignal)
   y1 er forrige utrekna utsignal
   y2 er utsignalet før det igjen
   osv.
   x0 er det siste målte spenning (innsignal)
   x1 er forrige innsignal
   x2 er innsignalet før det igjen
   osv.

4. Eit anna lavpass-filter er gitt ved differenslikninga

   y0 = b0•x0 + b1•x1 + b2*x2

   Lag eit program der du definerer tre konstantar b0=0.25, b1=0.50 og b2=0.25. Programmet skal så gå i ei sløyfe med lesing av ny verdi for innsignal x0, utrekning og utskrift av utsignal y0, flytting av verdien for x1 til x2 og verdien til x0 til x1. Programmet skal stanse når 15 verdiar er innlest.

5. I eit rekursivt filter er utsignalet også avhengig av tidlegare utsignal. Eit enkelt rekursivt filter er gitt ved likninga:

   y0 = b0•x0 - al*y1

   Lag eit program der du definerer to konstantar b0=0.2 og al=-0.8. Programmet skal gå i ei sløyfe med lesing av innsignal x0, utrekning og utskrift av utsignal y0, flytting av verdien for y0 til y1, til 20 verdiar er innlest. Prøv også med konstantane a1=-1.0 og a1=-1.2!

6. Lag eit program for utskrift av ein tabell over x og cos(x) når x varierer frå 0.00 til 2 pi i sprang på pi/10. Tabellen skal ha høveleg overskrift. Tips: Bruk x=n*pi/10 der n varierer frå 0 til 20.
Dette er eit eksempel på **sampling.** Tabellen viser 21 samplingar av cosinusfunksjonen.

7. Ei spenning varierer slik: u = 2.5*sin(2*pi*f*t). Vi måler spenninga ved tidspunkta t = nT der n = 0,1,2,.. Tida T mellom kvar måling kallar vi samplingsintervallet. Frekvensen f_S = 1/T kallar vi samplingsfrekvensen. Lag eit program for utskrift av ein tabell over tid t og spenning u i 1 sekund når frekvensen f = 2 Hz og samplingsfrekvensen er 20 Hz. Definer f og fs som konstantar, slik at det er lett å forandre dei. Prøv så programmet med andre frekvensar: f = 6 Hz,8Hz,10Hz, 12 Hz, 14 Hz. Når f passerer halve samplingsfrekvensen = Nyquistfrekvensen , ser vi at vi får verdiar som tyder på andre frekvensar enn dei vi har generert. Vi kallar dette for "Aliasing" Vi ungår "Aliasing" ved å halde oss under Nyquistfrekvensen.

8. Spenninga i oppgåve 7 blir sendt inn på filteret i oppgåve 4 . Lag eit program for utskrift av ein tabell over tid t, innspenning x0 = u og utspenning y0 i 1 sekund når frekvensen f = 2 Hz og samplingsfrekvensen er 20 Hz. Prøv så programmet med frekvensar f = 6 Hz, 8 Hz, 10 Hz, 12 Hz og 14 Hz. Studer amplituden på ut signalet i kvart tilfelle og finn dempningen (forhol det mellom amplitude ut og amplitude inn). Teikn skisse av dempningen som funksjon av frekvensen. Denne funk sjons-samanhengen kallar vi "frekvensresponsen" til filtret.

9. Lag eit C++-program for å løyse ei generell 2. gradslikning $ax^2 + bx + c = 0$. Algoritmen for dette kan vera slik:

Les inn koeffesientar a, b og c.
Rekn ut radikand R = $b^2 - 4ac$.
Dersom radikand er positiv så
    rekn ut og skriv ut x1 og x2 (bruk sqrt(R))
elles
    dersom radikand er lik 0 så
        rekn ut og skriv ut x1=x2
    elles
        rekn ut og skriv ut komplekse xl og x2
        (bruk sqrt(-R))

Tips: For komplekse tal reknar du ut realdel = -b/2a og imag=sqrt(-R)/2a og set saman utskrifta for eksempel slik:
```
cout<<"X1 = "<<realdel<<" + j "<<imag<<endl;
```

Legg så løysinga inn I Visual C++: Lag ein 2.gradsliknings-kalkulator.

10. Lag eit program for grafisk framstilling av cos(x) når x varierer frå 0.0 til 2pi. Opn ei fil GRAF.DAT for skriving, skriv overskrift "X COS(X)" på fila, skriv verdiar for x og cos(x) til fila. Køyr så programmet GRAFIKK.EXE. Tips: Bruk x=n*2pi/20 og lat n gå frå 0 til 20.

11. Skriv om programmet i oppgave 7 slik at funksjonen u blir FRAMSTILT GRAFISK når tida t varierer frå 0.0 s til 1.0 s. Studer den grafiske framstillinga når frekvensen varierer frå 2Hz til 18 Hz, og legg merke til "Aliasing" som oppstår når vi passerer Nyquist-frekvensen.

12. Skriv om programmet i oppgåve 8 slik at både innspenning x0 og utspenning y0 blir FRAMSTILT GRAFISK som funksjon av tida.

13. Studer eksempel 1 i kompendium "Litt om FOURIER-REKKER" av Norvald Midtun. Lag eit program for grafisk fram stilling av den periodiske funksjonen g(t) i intervallet t ε[-10,10] og fourier-rekka for g(t) der du tar med m ledd. Prøv med m = 1, 2, 3, 5 og 20.
Funksjonen g(t) deklarerer du slik:

```
double g(double t)
{
    If ((t<-5) || ((t>0) && (t<5))) return 3
    else return 0;
}
```

Algoritmen for hovedprogrammet kan bli slik:

```
 Opn fil GRAF.DAT for skriving
   Skriv overskrift til fil
   Lat n variere frå -100 til 100.
       Sett t = n/10
       Sett sum=3/2
       Lat i variere frå 0 til m-1     //m ledd i fourier-rekka
           sett k=2i+1                 //får då 1,3,5,7,...
           Sett sum=sum+(6/pi)*sin(k*pi*t/5)/k
       Skriv t, g(t) og sum til fil
   Lukk fil
   Køyr GRAFIKK.EXE
```

14.

Etter at vi har lært om tabellar (ARRAY), deklarerer vi digitaliserte spenningssignal-sekvensar som tabellar.
Innsignal x til eit filter og utsignal. y, deklarerer vi f.eks. som double x[20] og double y[20]. Ved kvar innlesing (eller måling), aukar indeksen n med 1.
I seinare oppgåver vil vi bruke desse notasjonene:

y[n] er det sist utrekna filtrerte signal (utsignal)
y[n-1] er forrige utrekna utsignal
y[n-2] er utsignalet før det igjen
osv.
x[n] er den siste målte spenning (innsignal)
x[n-1] er forrige innsignal
x[n-2] er innsignalet før det igjen
osv.

For eit generelt digitalt filter er samanhengen mellom innsignal og utsignal gitt ved ei såkalla differenslikning:

a0*y[n] + a1*y[n-1] + ... + aM*y[n-M] _=
b0*x[n] + b1*x[n-1] + ... + bN*x[n-N]

Lag eit program der du definer konstantane

b0=0.25;	bl=0.50;	b2=0.25;
a0=1;	a1=0.5;	a2=-0.5;

Programmet skal lese innsignal x[n] når n går frå 0 til 20, rekne ut og skrive ut utsignal y[n] =
(b0*x[n]+b1*x[n-1]+b2*x[n-2]-a1*y[n-1]-a2*y[n-2])/a0
Her vil datamaskinen huske alle tidlegare verdiar, så vi treng ikkje flytte verdiar slik som i oppgåve 4 og 5.
Men vi må modifisere likninga når n<2 så vi unngår tabellverdiar med indeks<0.
Etterpå skal datamaskinen framstille sekvensane x[n] og y[n] grafisk som funksjon av n.
Prøv også programmet med andre filterkonstantar.

15. Ein annan måte å finna utsignalet frå eit digitalt filter på, er å bruka den såkalla impulsresponsen:
Sender vi eit kort signal på 1 V ("enhetsimpuls") inn på filtret, får vi eit utsignal h[n] som vi kallar impulsresponsen.
Sender vi seinare inn eit generelt signal x[n] inn på filtret, så vil utsignalet y[n] vera gitt ved uttrykket

$$y[n] = \sum_{k=0}^{n} x[n-k] \cdot h[k]$$

I matematikken kallar vi ein slik operasjon ei folding av sekvensane {x} og {h}.
Lag eit program for slik folding når {h} er sekvensen gitt ved ein konstant tabell:

```
const double
h[]={0.25,0.5,0.25,0.1,0,-0.05,-0.1,-0.05,0,0,0,0,0,0,0,0,0,0,0,0}
```

Programmet skal lese innsignal x[n] for n frå 0 til 20, rekne ut og skrive ut utsignala y[n]. Etterpå skal innsignal og utsignal framstillast grafisk.