Datorlaborationer, Datorer och datoranvändning

Datorlaborationerna ger exempel på tillämpningar av det material som behandlas under kursen. Schema för laborationerna (tider och datorsalar) finns i kursprogrammet.

Laborationerna är obligatoriska. Det betyder att du måste bli godkänd på alla uppgifterna under ordinarie laborationstid. Om du skulle vara sjuk vid något laborationstillfälle så måste du anmäla detta till kursansvarig (`roger.henriksson@cs.lth.se`, 046–222 96 35) före laborationen. Om du varit sjuk bör du göra uppgiften på egen hand och redovisa den under påföljande laborationstillfälle. Det kommer också att anordnas en uppsamlingslaboration efter kursens slut.

Uppgifterna i laborationerna ska lösas individuellt. Regler för samarbete finns på nästa sida.

Varje laboration består av två delar: hemarbete och datorarbete. Innan du kommer till laborationen ska du ha förberett dig genom att ha gjort uppgifterna under hemarbete samt läst igenom vad som står under datorarbete. Du ska också ha gått igenom kontrollfrågorna.

Om du hittar någonting i uppgifterna eller andra anvisningar som är felaktigt eller oklart så är vi tacksamma om du meddelar dina synpunkter till `roger.henriksson@cs.lth.se`.

Du måste för varje laboration se till att laborationsledaren noterar dig som godkänd på listan på sista sidan i häftet.

I början av varje laboration kommer laborationsledaren att kontrollera att du har förberett dig. Kontrollen görs genom att du får en skrivning med fyra frågor som valts bland kontrollfrågorna. Du måste besvara minst tre av frågorna korrekt för att du ska få genomföra laborationen.

Svaren till kontrollfrågorna kommer att finnas i laborationshandledningen eller i annat material som delas ut eller hänvisas till. I några fall kan du behöva tänka eller reflektera lite grand själv med utgångspunkt från materialet.

Riktlinjer för inlämningsuppgifter och laborationsuppgifter

I studiehandboken (Gemensamma föreskrifter och information, avsnitt 1.23) finns följande föreskrifter:

- Inlämningsuppgifter skall fullgöras individuellt om det inte särskilt anges att de skall fullgöras i grupp.
- Vid arbete i grupp bestämmer ansvarig lärare om gruppindelningen och ändringar av denna. Arbetet skall utföras av dem som ingår i gruppen.
- Det är tillåtet att diskutera uppgifterna och tolkningen av dessa med utomstående på ett allmänt plan men inte att få hjälp med de konkreta lösningarna.
- Det är inte tillåtet att kopiera annans eller annan grupps lösningar helt eller delvis. Det är inte heller tillåtet att kopiera från exempelvis litteratur eller Internet. Vid citat skall källan tydligt anges.
- Väsentlig hjälp, av annan än lärare på kursen, för att genomföra en uppgift skall redovisas i redogörelsen eller på annat tydligt sätt. Detsamma gäller om man använt någon annan form av hjälpmedel som läraren inte kan förutsättas känna till.
- Institutionerna kan komplettera dessa regler skriftligen i samband med kursstarten, exempelvis i ett kursprogram.

Vid institutionen för datavetenskap gäller även följande kompletteringar/förtydliganden av reglerna:

- Reglerna om arbete i grupp ovan tillämpas för alla obligatoriska moment som utförs i grupp det vill säga även laborationer och projektarbeten.
- Då arbete görs i grupp skall alla gruppdeltagare delta i arbetet.
- Hjälp från annan med handhavandet av apparatur, utnyttjandet av datorsystem och givna datorprogram behöver inte redovisas.

Kontakta ansvarig lärare om du är osäker på om viss hjälp är tillåten eller inte!

Institutionen tillämpar dessa riktlinjer på alla kurser. Om vi är övertygade om att fusk skett så överlämnar vi ärendet till universitetets disciplinnämnd för vidare åtgärd. Finner disciplinnämnden de studerande skyldiga är påföljden upp till sex månaders avstängning från universitetet och högskolan.

Laboration 1 — Unix

Mål: Du ska befästa och utöka de kunskaper om Unix som du fick under datastugan i introduktionsveckan. Du blir inte Unix-expert på en laboration, och det behöver du inte heller vara. Men det är viktigt att du är van att arbeta med Unix; det kommer att underlätta dina studier i fortsättningen.

Hemarbete

H1. Läs igenom uppgifterna under rubriken Datorarbete och titta igenom de avsnitt av kompendiet ”Introduktion till LTH:s Unixdatorer” som det hänvisas till i uppgifterna. I hänvisningarna förkortas namnet på detta kompendium till ILU.

Kontrollfrågor

K1. Hur gör man för att byta lösenord på studentdatorerna?
K2. Vad betyder kommandot `pwd`?
K3. Vad är en kommandotolk? Vilken kommandotolk används på Linuxdatorerna i E-huset?
K4. Vad heter skrivbordsmiljön som används på Linuxdatorerna i E-huset?
K5. Hur återkallar man ett tidigare givet kommando?
K6. Hur kan man få hjälp om användningen av ett kommando, förutsatt att man vet namnet på kommandot?
K7. Hur skriver man ut en innehållsförteckning över den aktuella katalogen?
K8. Vilka är kommandona för att skapa respektive ta bort kataloger?
K9. Hur byter man aktuell katalog?
K10. Vad betyder tecknen `?` och `*` när man skriver dem på en kommandorad?
K11. Vad betyder tecknen < och > när man skriver dem på en kommandorad?
K12. Förklara kort hur systemet med åtkomsträttigheter av filer och kataloger fungerar.
K13. Vilket kommando utnyttjar man om man vill titta på innehållet i en fil en sida i taget?
K14. Antag att programmet `prog` producerar många sidor utskrift. Hur gör man för att titta på utskriften en sida i taget?
K15. Vad är en process?
K16. Hur avbryter man ett exekverande program?

Datorarbete

D1. Logga in på datorn. Använd det användarnamn och det lösenord som du tidigare har kvitterat ut (ILU 1.3).

D2. Fönsterhantering (ILU 1.4). På Linuxdatorerna i E-huset används Linuxdistributionen Ubuntu och skrivbordsmiljön Unity . Om du inte redan har gjort det under datorstugan: ägna några minuter åt att bekanta dig med Unity.

D3. Editering av kommandoraden (ILU 3.2), enkla Unix-kommandon.

a) Öppna ett kommandofönster (Terminal).
b) Skriv några enkla Unix-kommandon, till exempel `date` och `cal`.
c) Skriv avsiktligt fel och rätta felet. Prova specialtecknen för att radera enstaka tecken på kommandoraden och för att radera hela raden.
d) Prova också specialtecknen för att få tillbaka tidigare kommandon. Återkalla kommandot för utskrift av datum och utför det på nytt.

e) Skriv ut hjälptexten (”man-sidan”) för `date`-kommandot (ILU 1.8). Bläddra framåt och bakåt i texten.

f) Använd kalenderprogrammet (`cal`) för att ta reda på vilken veckodag du är född.

D4. Kommandon för att hantera filer och kataloger (ILU 1.7, 2.2, 2.7–2.10).

a) Skriv ut en innehållsförteckning över din hemkatalog. Skriv ut en förteckning där också filerna vars namn börjar med punkt skrivs ut.

b) Gå till katalogen */usr/local/cs/dod/lab2/metool/src/metool*. Skriv ut en innehållsförteckning över katalogen. Skriv ut en förteckning över de filer vars namn innehåller strängen *Statement*.

c) Prova hur filnamnskomplettering fungerar. Skriv `less R` och tryck på TAB. Datorn fyller i tecken i filnamnet så länge de är unika (nu står det `less Re` på kommandoraden). Det finns mer än en fil vars namn börjar med *Re*. Tryck på TAB en gång till så får du en lista över dessa filer. Skriv `a` och tryck på TAB; datorn fyller i till det unika filnamnet *ReadStatement.java*. Tryck på RETURN för att titta på filen.

d) Gå till din hemkatalog och skapa en katalog för de filer som används i denna laboration. Katalogen ska heta *lab1* och vara en underkatalog till en katalog *dod*, där du ska spara allt som rör kursen Datorer och datoranvändning. Du ska i fortsättningen skapa en ny katalog för varje datorlaboration som du gör. Använd följande kommandon:

```
cd          // gå till din hemkatalog
mkdir dod   // skapa katalogen dod i hemkatalogen
cd dod      // gå till katalogen dod
mkdir lab1  // skapa katalogen lab1
cd lab1     // gå till lab1
```

e) Kopiera filen */usr/local/cs/dod/lab1/example.txt* till katalogen *lab1*. Skriv ut filen på skärmen med en sida i taget.

f) Undersök hur mycket utrymme du har tillgängligt för att lagra filer.

g) Tag reda på hur stor filen *example.txt* är. Komprimera därefter filen och tag reda på storleken hos den komprimerade filen. Återställ sedan filen till sitt ursprungliga utseende.

h) Skriv ut de rader i filen *example.txt* som innehåller ordet Unix. Kommandot för att leta i en fil heter `grep`.

i) Samma som uppgift h, men koppla om utskriften så att den hamnar i en fil med namnet *unix.txt*. Skriv ut denna fil på skärmen.

j) Räkna (med ett kommando) antalet rader i filen *unix.txt*. Du har nu räknat antalet rader som innehåller ordet Unix i filen *example.txt*.

k) Tag bort filen *unix.txt*.

l) Gör samma sak som i uppgift i–k utan att använda en temporär fil. Koppla i stället ihop kommandona med en pipe.

D5. Editering av text. I Ubuntu finns flera editorer, till exempel `nano` (enkel, terminalbaserad), `gedit` (enkel, fönsterbaserad) och `emacs` (avancerad). Du får naturligtvis använda vilken editor du vill, men här ska du testa `gedit`. Gör gärna om uppgifterna nedan i `emacs` senare, på egen hand.

a) Starta `gedit` och läs in filen *example.txt* genom att i terminalfönstret skriva `gedit example.txt &`.

b) Utnyttja musen och piltangenterna för att flytta textmarkören. Ändra textbuffertens innehåll genom att ta bort tecken och skriva in tecken. Spara innehållet i textbufferten på filen.

c) Kontrollera att filen *example.txt* har ändrats.

d) Dela en rad i två rader. Sätt ihop raden igen.

e) Lägg in några tomma rader, tag sedan bort dem igen.

f) Utnyttja rullningslisten för att flytta dig i texten. Gå till början av texten. Gå till slutet av texten. Gå till rad 43 i texten.

g) Markera ett textblock genom att trycka på vänster musknapp och dra markören eller genom att trycka vänster och sedan skift-vänster. Markera ett ord och en hel rad.

h) Kopiera ett markerat textblock till en annan plats i bufferten. Flytta sedan ett markerat textblock.

i) Gå till början av filen och leta upp den första förekomsten av ordet Unix. Leta sedan upp nästa förekomst, osv. Byt sedan alla Unix mot Xinu.

D6. Hantering av processer (ILU 3.2, 3.9).

a) Skriv kommandot `xeyes` i terminalfönstret. Flytta musen så ser du att ögonen följer musmarkören. Notera att man inte kan fortsätta att skriva kommandon i fönstret eftersom det är låst av `xeyes`-programmet.

b) Skriv CONTROL-C i kommandofönstret för att avbryta `xeyes`-programmet. Programmets fönster försvinner när man avbryter programmet.

c) Skriv nu `xeyes &` i terminalfönstret. `&`-tecknet betyder att programmet ska köras som en självständig process som inte är kopplad till terminalfönstret. Nu kan man alltså fortsätta att skriva kommandon i terminalfönstret. Avsluta `xeyes` genom att klicka på krysset i fönstrets titelrad.

d) Man kan tillfälligt avbryta exekveringen av ett program med CONTROL-Z. Skriv `xeyes` och sedan CONTROL-Z. Notera att programmet nu inte är aktivt (ögonen följer inte musmarkören). Med kommandot `fg` återupptar man exekveringen igen. Om man i stället ger kommandot `bg` återupptar man exekveringen ”i bakgrunden”, precis som om man hade startat programmet med `xeyes &`.

D7. Glöm inte att logga ut innan du lämnar datorn!

Laboration 2 — Lågnivåprogrammering

Mål: Du ska lära dig grunderna i hur en dator fungerar på maskinspråksnivå och träna på att skriva små program i assemblerpråk.

Hemarbete

H1. Titta igenom overheadbilderna till veckans föreläsning.

H2. Läs igenom häftet ”ME – en dator”. Om du inte har förkunskaper i programmering behöver du inte läsa igenom avsnitt 2.2 och 2.3; de avsnitten är överkurs.

H3. Skriv ett ME-program som löser följande problem: läs in två tal, skriv ut talens summa, skillnad, produkt och kvot.

H4. Skriv ett ME-program som först läser in ett tal `n`. Därefter ska `n` tal läsas och talens summa och produkt beräknas och skrivas ut.

H5. Skriv ett ME-program som läser in ett antal tal och räknar hur många av talen som är större än noll och hur många som är mindre än noll. När en nolla läses in ska inläsningen avslutas och de båda antalen skrivas ut.

H6. Bara för dig som har förkunskaper i programmering: Skriv ett ME-program som läser in ett `n`-värde och sedan `n` tal. Skriv ut talen i omvänd ordning.

H7. Bara för dig som har förkunskaper i programmering: Skriv ett ME-program som läser in ett `n`-värde och sedan `n` tal. Sortera talen i växande ordning och skriv ut dem.

Kontrollfrågor

K1. Översätt det binära talet `1001011` till hexadecimal och till decimal form.
K2. Vad är anledningen till att man i datorer använder det binära talsystemet istället för det decimala?
K3. Vilket är det största talet som kan representeras med två decimala siffror? Hexadecimala siffror? Binära siffror? Uttryck svaren decimalt.
K4. PC kan betyda ”Personal Computer”, men det finns också en komponent i en processor med samma förkortning. Hur uttyds PC i en processor?
K5. Vilka delar består en processor av?
K6. Redogör för begreppen assemblerspråk och maskinspråk.
K7. Vad kan det finnas för fördelar med att programmera i högnivåspråk, som exempelvis Java, istället för i assemblerspråk?
K8. Redogör för datorers instruktionscykel.
K9. Vilka är parametertyperna i ME-datorn? Förklara dem kortfattat.
K10. Förklara vad följande ME-instruktioner gör:

```
read  r1
add   r1,1,r1
add   r1,m(1),m(2)
jpos  r1,back
```

K11. Skriv om följande Javasatser med ME-instruktioner (på skrivningen före laborationen kan det komma ett annat exempel, utan `if`-, `while`- eller `for`-satser):

```
int x = 1;
int y = 2;
int z = 2 * (x + 7) - 8 / y;
```

K12. Vad skrivs ut i följande ME-program (ett annat exempel kan komma på skrivningen):

```
main: move   1,m(0)
loop: sub    m(0),5,r1
      jpos   r1,ends
      mul    m(0),2,m(0)
      jump   loop
ends: print  m(0)
      stop
```

K13. Laborationsledaren kontrollerar att du har skrivit programmen i hemuppgifterna H3, H4 och H5.

Datorarbete

D1. Gå till katalogen *dod* som du skapade i laboration 1. Skapa en ny katalog med namnet *lab2* och gå till denna katalog.

D2. Ge följande kommando för att starta programmet `METool`:

```
java -jar /usr/local/cs/dod/lab2/metool/METool.jar &
```

D3. Skriv in programmet från uppgift H3 i en fil *h3.mep*. Ladda in det i METool, kompilera och rätta eventuella fel. Testkör programmet, både genom att köra från början till slut och genom att köra stegvis.

D4. Skriv in och testa programmen från uppgift H4 och H5.

D5. Du som löst uppgift H6 eller H7: skriv in och testa programmen från dessa uppgifter.

Laboration 3 — LaTeX

Mål: Du ska lära dig grunderna i LaTeX och tillämpa dina kunskaper på ett exempel.

Hemarbete

H1. Titta igenom overheadbilderna till veckans föreläsning.

H2. Läs igenom kompendiet "Att skriva rapporter med LaTeX", åtminstone så mycket så att du blir bekant med vad man kan göra med LaTeX. Du behöver inte försöka lära dig några detaljer.

H3. Med början på nästa sida finns ett exempel på en rapport som är producerad med LaTeX. Studera rapporten och försök komma på vilka kommandon som behövs för att få texten att se ut som den gör. Markera i rapporten vilka kommandon du tänker använda.

Kontrollfrågor

K1. Laborationsledaren kontrollerar att du gått igenom exempelrapporten och markerat vilka LaTeX-kommandon du ska utnyttja för att formatera texten.

Datorarbete

D1. Skapa en ny katalog med namnet *dod/lab3* och gå till denna katalog.

D2. Kopiera filen */usr/local/cs/dod/lab3/rapportmall.tex* till din katalog. Ge den kopierade filen namnet *rapport.tex*. Filen innehåller en LaTeX-mall för rapporter, liknande den mall som beskrivs i avsnitt 2.2 i LaTeX-kompendiet och i overheadbilderna.

D3. I filen */usr/local/cs/dod/lab3/rapporttext.txt* finns texten till rapporten som beskrivs i uppgift H3, utan några LaTeX-kommandon och utan några figurer. Lägg in innehållet i denna fil i din *.tex*-fil, mellan `\begin{document}` och `\end{document}`.

D4. Starta Texmaker med kommandot `texmaker &` (eller `texmaker filnamn.tex &`).

D5. Lägg in lämpliga LaTeX-kommandon i filen så att rapporten får (åtminstone ungefär) det utseende som beskrivs i uppgift H3. Se till att styckeindelningen och rubrikerna blir korrekta i hela dokumentet innan du ger dig på resten, till exempel de matematiska formlerna.

Arbeta stegvis: ändra litet, klicka på pilen till vänster om Quick Build så körs pdfLaTeX, titta på resultatet, ändra litet till, osv.

Bilderna som ska inkluderas i dokumentet finns i filerna *nrbild.pdf* och *konvbild.pdf* i katalogen */usr/local/cs/dod/lab3/*, programkoden finns i filen *NewtonRaphson.java* i samma katalog.

D6. Skriv ut rapporten på skrivaren när du är nöjd med dokumentets utseende.

D7. Om du har tid: prova sådana möjligheter i LaTeX som du inte har behövt använda tidigare: listor av olika slag, innehållsförteckning, mera avancerade formler, osv.

Newton-Raphsons metod

Per Holm

3 september 2013

1 Numeriska metoder

En *numerisk metod* är en metod med vars hjälp man kan finna en approximativ lösning till ett matematiskt problem. Lösningen ges i numerisk form dvs i form av ett antal talvärden. Till exempel kan man med en numerisk metod räkna ut att roten till ekvationen $x^2 = 3$ är $1.73205\ldots$, men inte få fram den analytiska lösningen $\sqrt{3}$.

I många problem är man hänvisad till numeriska metoder. Det kan bero på att problemet inte har någon analytisk lösning eller att det skulle vara alltför arbetsamt att få fram denna.

I denna rapport studeras en numerisk metod för att lösa ekvationer med en obekant, nämligen Newton-Raphsons metod.[1]

2 Newton-Raphsons metod

2.1 Teori

Vi söker en rot α till ekvationen $f(x) = 0$. Vi antar att vi vet att roten ligger i närheten av talet x_0 och ska försöka förbättra denna approximativa lösning. Vi kan bilda nya approximationer $x_1, x_2, x_3, \ldots$ med följande formel:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{1}$$

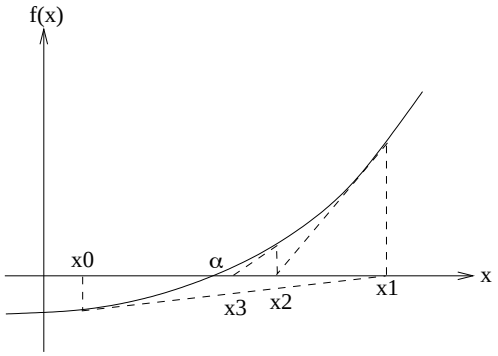
Metoden åskådliggörs geometriskt i figur 1. Tangenten i punkten $(x_0, f(x_0))$ skär x-axeln för $x = x_1$, tangenten i punkten $(x_1, f(x_1))$ skär x-axeln för $x = x_2$, osv. Av figuren förstår vi att $x_k \to \alpha$ då $k \to \infty$.

Newton-Raphsons metod kan härledas genom att man serieutvecklar funktionen $f(x_0 - \epsilon)$ och trunkerar utvecklingen efter den linjära termen. Ur serieutvecklingen kan man också få ett uttryck på feltermen.

2.2 Exempel

Ekvationen $e^{-x} = \sin x$ har en rot nära 0.6. Bestäm denna rot med Newton-Raphsons metod.

[1] Metoden kallas ibland Newtons metod.



Figur 1: Newton-Raphsons metod

Här är $f(x) = e^{-x} - \sin x$, $f'(x) = -e^{-x} - \cos x$. Om vi räknar med 9 decimaler så får vi:

x	$f(x)$	$f'(x)$	$f(x)/f'(x)$
0.6	-0.015830837	-1.374147251	0.011510481
0.588479519	0.000073820	-1.386956425	-0.000053224
0.588532743	0.000000001	-1.386897331	-0.000000001
0.588532744			

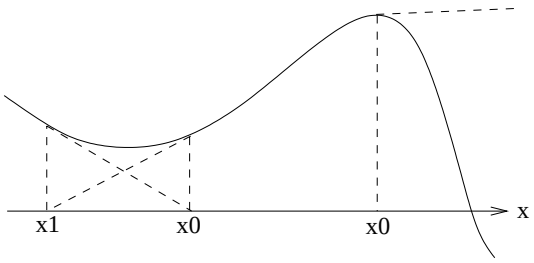
dvs $\alpha = 0.588532744$.

2.3 Konvergens

Newton-Raphsons metod konvergerar i allmänhet *kvadratiskt* mot den sökta roten, vilket innebär att antalet korrekta siffror i svaret fördubblas i varje iteration. Men det finns fall då konvergensen blir sämre eller då metoden inte alls konvergerar, nämligen då man råkar ut för en derivata som är nära noll. Två sådana fall illustreras i figur 2.

3 Newton-Raphsons metod i Java

Det är inte helt enkelt att implementera Newton-Raphsons metod i ett program, åtminstone inte om man kräver att metoden ska fungera i alla upptänkliga fall. I



Figur 2: Konvergensproblem

metoden `solve` som visas nedan finns bara den grundläggande algoritmen med: vi har inte tagit hänsyn till undantagsfall som finns eller fel som kan inträffa, till exempel att derivatan `fprim(x0)` blir nära noll (se avsnitt 2.3).

```
class NewtonRaphson {
    private static double f(double x) {
        return ...;
    }

    private static double fprim(double x) {
        return ...;
    }

    public static double solve(double x0, double eps) {
        double x1 = x0;
        do {
            x0 = x1;
            x1 = x0 - f(x0) / fprim(x0);
        } while (Math.abs(x1 - x0) > Math.abs(x1) * eps);
        return x1;
    }
}
```

Laboration 4 — Internet inuti

Mål: Du ska med hjälp av olika Unix-program studera hur Internet fungerar ”inuti”. Laborationen ger en orientering om fundamentala Internetprotokoll som Telnet, HTTP och SMTP samt visar hur data finner vägen genom nätet.

Hemarbete

H1. Läs noga igenom hela laborationen. Du bör därefter kunna svara på en del av frågorna i texten. De övriga frågorna ska du svara på under laborationen.

H2. Läs igenom avsnittet om e-post (2.5) i kompendiet Introduktion till LTH:s Unixdatorer.

H3. Denna hemuppgift ska utföras vid dator. Starta en webbläsare och surfa till

`http://www.w3schools.com/html/`

eller till någon liknande handledningssida och orientera dig om möjligheterna i språket HTML. Om du är nybörjare på HTML bör du sätta ett bokmärke på denna sida så att du lätt kan hitta informationen när du behöver den.

Kontrollfrågor

K1. Uttyd följande förkortningar: DNS, HTTP, FTP, SMTP, WWW, TCP, POP.
K2. Vad är en IP-adress? Hur ser en typisk sådan ut?
K3. Hur gammalt är Internet? Hur gammal är webben? E-posten?
K4. Var startades Internet? Varför?
K5. Vad är Ethernet?
K6. Vad gör en namnserver?
K7. Ge några exempel på HTML-taggar och vad de används till.
K8. Vilka protokoll används vid e-postkommunikation?
K9. Vilket protokoll används för att skicka webbsidor?
K10. Av vilka delar utgörs en URL?
K11. Hur hög är en typisk överföringshastighet vid kommunikation via ADSL?
K12. Vad gör en router?
K13. Vilken är skillnaden mellan telnet och ssh?

Datorarbete

Webbservern, webbsidor

Vi ska börja med att titta närmare på vad som händer när man använder WWW. Först ser vi på servern som tillhandahåller informationen.

D1. En URL (Uniform Resource Locator) är en adress till ett dokument på webben. Betrakta följande URL:

`http://users.student.lth.se/dat14xyn/dod.html`

Denna URL refererar till filen *dod.html* på datorn `users.student.lth.se`. Filerna är organiserade annorlunda när de omnämns i URL-er än i det lokala filsystemet. På datorerna i domänen `student.lth.se` motsvarar ovanstående URL filen

~dat15xyn/public_html/dod.html

Filen ligger alltså i en katalog *public_html* som i sin tur ligger i din hemkatalog.

Skapa katalogen som behövs (*dat15xyn* ska naturligtvis bytas mot ditt eget användarnamn). Se till att katalogen får rättigheterna `r-x` för användarna ”others”.

D2. Du ska nu tillverka en webbsida för URL-en ovan. Det enklaste sättet att skapa en webbsida är att utgå från en existerande sida som liknar den man vill skapa och modifiera den.

a) Gå till kursens hemsida:

`http://cs.lth.se/EDA070/`

Under laboration 4 finns en länk till en webbsida med namnet *dod.html*. Gå till denna sida och välj ”Save As” i File-menyn. Se till att filen hamnar i katalogen du nyss skapade och gör den läsbar för användarna ”others”. Öppna filen i en editor och jämför med vad som visas i en webbläsare.

b) Som den mesta informationen på WWW så är sidan skriven i HTML (Hypertext Markup Language). HTML är ett sätt att beskriva text med struktur, till exempel styckeindelning och punktlistor, liknande LaTeX fast mera primitivt. Du ser i texten att det finns insprängda direktiv som beskriver textens formatering (fetstil, rubriker och liknande).
Ändra texten litet grann efter behag. Huvudsaken är att din nya sida är annorlunda än den ursprungliga. Spara sedan din nya version.

c) Ladda in din sida i webbläsaren och betrakta resultatet.
Texten skickas över nätet från servern till klienten i HTML-form och det är alltså upp till klienten (webbläsaren) att bestämma exakt hur formgivningen ska göras.

d) För att se en annan formgivning ska du också prova den textbaserade webbläsaren `lynx`. Ge kommandot (återigen med `dat14xyn` utbytt mot ditt eget användarnamn):

```
lynx http://users.student.lth.se/dat14xyn/dod.html
```

Du ser nu samma sida, men annorlunda formgiven. Tryck på `Q` för att avsluta `lynx`.

DNS, ARP, vägval och SAP

Denna del av laborationen handlar om hur datapaket hittar fram till sin destination på Internet. Detta kallas vägval eller på engelska routing. Först måste vi förstå hur datorer identifieras.

Det finns två sätt att hänvisa till en dator på Internet: med datorns namn, till exempel `lo-5.student.lth.se`, eller med dess IP-nummer, till exempel `130.235.35.193`. Det viktiga för funktionaliteten är IP-numret. Namnet finns bara för att det är lättare för människor att komma ihåg.

När ett program ska kontakta en dator med ett visst namn måste det börja med att ta reda på vilket IP-nummer som namnet motsvarar. För detta ändamål finns en så kallad *namnserver*[1] (Domain Name Server, DNS) för varje domän. Det finns till exempel en namnserver för domänen `student.lth.se` och en annan för `cs.lth.se`. Namnservern innehåller en stor tabell där varje datornamn associeras till ett unikt IP-nummer.

D3. Man kan ställa frågor till namnservern med hjälp av ett program kallat `host`. Vanligen ger man ett datornamn eller ett IP-nummer som argument till programmet och som svar får man båda uppgifterna. Man kan också ge fler parametrar för att ta reda på information om hela domäner. Till exempel kan man ta reda på namnet på e-postservern (”mail exchanger”) till en domän genom att före domännamnet ge optionen `-t MX`.

[1] Läs mera om namnservrar på `computer.howstuffworks.com/dns.htm` eller på `en.wikipedia.org/wiki/Domain_name_system`

a) Vilket IP-nummer har datorn du sitter vid?
b) Vad heter datorn med IP-numret 193.235.142.142?
c) Vilken e-postserver hanterar post från `student.lth.se`?

På kablar som förbinder datorer (fiber eller koaxialkablar) utnyttjas ett ”lågnivåprotokoll” som kallas *Ethernet*. Ethernet har också ett speciellt sätt att referera till maskiner, nämligen med det så kallade Ethernetnumret eller den ”fysiska adressen”.

Ett Ethernetnummer består av sex bytes (till skillnad från ett IP-nummer som består av fyra bytes). Numret är fysiskt knutet till den nätverkskoppling datorn är utrustad med, till exempel har varje nätverkskort ett unikt sådant nummer inbränt i ett ROM på kortet. Förutom kopplingen mellan datornamn och IP-nummer måste det finnas en motsvarande koppling mellan IP-nummer och Ethernetnummer för att kommunikationen ska fungera. Protokollet som upprätthåller dessa kopplingar heter *Address Resolution Protocol* (ARP).

Varje dator har en dynamisk tabell med kända Ethernetnummer och IP-nummer. Om ett Ethernetnummer till en viss maskin saknas då man försöker kontakta den görs en särskild uppslagning över nätet och tabellen kompletteras med information.

Data som skickas över nätet passerar olika maskiner på vägen till sin destination. Varje nod (maskin) i nätet har information om vilken väg data avsedda för andra maskiner ska skickas (”routas”) för att nå fram på snabbaste sätt.

D4. Man kan spåra vilken väg data tar genom nätet med kommandot `traceroute`. Använd detta kommando för att ta reda på vilken väg information skickas från din maskin till LTH (`www.lth.se`) och till några andra datorer.

Flera program på en maskin kan kommunicera samtidigt med olika datorer på nätet, till exempel kan man surfa och skicka e-post samtidigt. Det som gör detta möjligt är att varje protokoll är associerat med en särskild *Service Access Point* (SAP), som är ett positivt heltal. På Internet kallas SAP vanligen *portnumret*.

I Unix finns i filen */etc/services* en lista på Internetprotokoll och deras motsvarande portnummer.

D5. Protokollet NTP (Network Time Protocol) används mellan maskiner i ett nätverk för att utväxla information om vad klockan är. Många andra protokoll och applikationer är beroende av att datorernas klockor är synkroniserade med varandra, och detta sköts alltså om av NTP.

a) Vilket portnummer är associerat med NTP?
b) Vilket protokoll använder port nummer 79?

Telnet och HTTP

Ett av de äldsta Internetprotokollen kallas *Telnet*. Telnet är namnet både på protokollet och på ett program som används för att öppna vanliga datakommunikationskanaler (så kallade sockets) mellan olika accesspunkter på två maskiner. Om man känner till ett annat protokoll i detalj kan man via Telnet ”manuellt” prata med en server på en annan maskin, till exempel med webbservern.

Protokollet som används för att skicka webbsidor kallas för HTTP (Hypertext Transfer Protocol). Som bekant behöver man inte känna till detta protokoll för att kunna utnyttja WWW. De klientprogram som finns för att man ska kunna surfa, exempelvis Firefox, Opera eller MS Internet Explorer, implementerar protokollet åt användaren. Nu ska vi dock ta en titt på det viktigaste kommandot i HTTP, nämligen `GET`. Varje gång man beordrar en webbläsare att hämta en webbsida utför webbläsaren detta kommando.

D6. Börja med att ansluta till student-webbservern genom att ge kommandot

```
telnet users.student.lth.se 80
```

Det första argumentet till `telnet` är namnet på datorn man ansluter till. Talet 80 är portnumret för protokollet HTTP. Telnet svarar med:

```
Trying 130.235.20.66...
Connected to bertil.ddg.lth.se.
Escape character is '^]'.
```

och stannar och väntar på kommandon.

D7. Begär nu webbsidan du tillverkade tidigare genom att ge kommandot

```
GET http://users.student.lth.se/dat14xyn/dod.html HTTP/1.0
<blank rad>
```

Du bör känna igen svaret.

Observera att servern kopplar ner förbindelsen om man inte skriver något kommando inom några sekunder. Det kan därför vara bra att skriva kommandot först på kommandoraden och sedan markera och kopiera texten så att du sedan snabbt kan klistra in den innan servern hinner koppla ned.

D8. Ett annat protokoll är *daytime*, som har portnumret 13. Pröva att med `telnet` ansluta till denna port på någon dator som stöder daytime-protokollet (en lista över sådana datorer finns på `http://tf.nist.gov/tf-cgi/servers.cgi`).

Hur protokollen ser ut i detalj bestäms i dokument kallade RFC-er (RFC står för Request for Comments). Daytime-specifikationen i RFC 867 är extremt kort (leta upp den på nätet och titta på den); de flesta andra protokoll är betydligt mera omfattande.

D9. Om man utelämnar portnumret på kommandoraden kommer `telnet` att ansluta till den förvalda porten 23, där en telnet-server svarar. Det som då händer är att användaren får logga in och därefter startar servern en kommandotolk på målmaskinen. Eftersom all kommunikation över telnet sker utan kryptering (även lösenord skickas okrypterade) ska man *aldrig* använda telnet för att köra på andra datorer utan i stället `ssh` (Secure Shell). `ssh` fungerar som telnet men kommunicerar krypterat och använder portnummer 22. Om du inte har testat att använda ssh tidigare mot andra datorer, gör det nu. Använd login-datorn `login.student.lth.se` (denna dator ska man alltid använda för inloggning utifrån).

SMTP – Simple Mail Transfer Protocol

Ända sedan starten av Internet har det funnits ett protokoll för att skicka e-post. Protokollet, som används än idag, heter *Simple Mail Transfer Protocol* (SMTP, RFC 2821).

Alla e-postprogram implementerar SMTP eller delar av det, på liknande sätt som en webbläsare förstår sig på HTTP. I protokollet specificeras ett antal kommandon, av vilka de viktigaste finns i tabell 1.

Efter varje kommando svarar servern med en statuskod. Den är 250 om kommandot har utförts korrekt, annars ges ett felmeddelande i form av någon annan kod som specificeras av protokollet.

SMTP använder vanligen portnummer 25 för sin kommunikation. När man ansluter till en e-postserver börjar servern alltid dialogen med att presentera sig. Man svarar därefter med `EHLO` som i tabellen på följande sida. Därefter skickar man ett brev genom att ge kommandona `MAIL`

`HELP`	Ger hjälp om de övriga kommandona.
`EHLO userdomain`	Hälsningsfras[a] som man alltid inleder konversationen med efter att ha anslutit sig till en SMTP-server. Parametern `userdomain` är den Internetdomän man ansluter ifrån, exempelvis `student.lth.se`.
`MAIL FROM: <address>`	Inleder en sändning av ett brev. Parametern `address` är avsändarens (din) e-postadress, till exempel `dat14xyn@student.lu.se`
`RCPT TO: <address>`	Anger mottagaren av det brev man vill sända. Aven i detta fall är parametern address en fullständig e-postadress, till exempel `dic14zze@student.lu.se`
`DATA`	Markerar starten på texten i brevet. Efter att man har gett detta kommando skriver man texten och avslutar med en ensam punkt på en rad. Då sänds brevet iväg.
`QUIT`	Avslutar förbindelsen med e-postservern.

[a] `EHLO` hette tidigare `HELO`. Båda kommandona står för ”hello”.

Tabell 1: SMTP-kommandon.

`FROM`, `RCPT TO` och `DATA` tillsammans med lämpliga parametrar och data. Slutligen avslutar man med kommandot `QUIT`, varvid servern stänger förbindelsen.

Allt detta gör alltså alla e-postprogram, som `exmh`, `pine` eller `Eudora`, åt användaren varje gång ett brev skickas. Nu ska du pröva att göra det manuellt.

D10. Använd `telnet` (port 25) för att ansluta till LTH:s e-postserver (`mail.lth.se`) eller till den server som du identifierade i uppgift D3.

D11. Utnyttja protokollspecifikationen för att skicka ett brev till dig själv. När brevet anländer, studera resultatet i ett e-postprogram.

D12. Du ser i det mottagna brevet att mottagaradressen inte är specificerad, trots att du angav den med ett kommando till SMTP. Detta beror på att kommandot bara är en instruktion till SMTP som inte automatiskt blir ett fält i brevet. Man kan få detta fält specificerat i brevet genom att först i `DATA`-blocket skriva

```
To: dic14zze@student.lu.se
```

Här bör man också ange ämne, till exempel

```
Subject: Ett litet test
```

Dessa rader tolkas på speciellt sätt av mottagarens e-postprogram och sätts in överst i avsedda fält. Testa att detta fungerar!

D13. Denna avslutande del som handlar om SMTP har för den uppmärksamme avslöjat ett säkerhetsproblem med protokollet. Fundera över följande frågor:

a) Vari består nämnda säkerhetsproblem och vad kan det ha för konsekvenser?

b) Hur kan man som e-postanvändare skydda sig mot problemet?

När du är klar

Diskutera igenom med din handledare vad du har lärt dig och visa att du har svarat på samtliga frågor.

Laboration 5 — Matlab

Mål: Du ska bekanta dig med Matlab. I laborationen visar vi bara de mest elementära delarna av Matlab — det finns massor av färdiga paket och funktioner för olika användningsområden. En del av dessa kommer du att träffa på i senare kurser.

Lunds Universitet har avtal med MathWorks så att studenter har gratis tillgång till Matlab. Du kan ladda hem Matlab till din egen dator från `http://program.ddg.lth.se`. Alternativt kan du använda Octave, `www.octave.org`.

Hemarbete

H1. Läs igenom overheadbilderna till veckans föreläsning. Det finns gott om Matlab-introduktioner på webben; leta själv om du är intresserad.

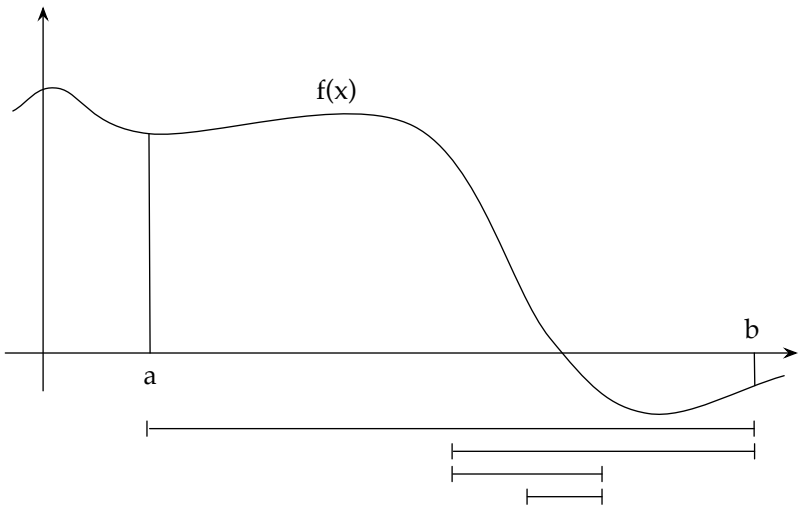
H2. Lös (för hand) följande ekvationssystem:

$$\begin{aligned} 5x - 3y &= 1 \\ x + 2y &= 8 \end{aligned}$$

H3. En enkel och säker metod att hitta nollställen till en funktion f är intervallhalvering. Man utgår från ett startintervall $[a, b]$ sådant att $f(a)$ och $f(b)$ har olika tecken. I intervallet finns då med säkerhet (minst) ett nollställe till $f(x)$. Målet är att göra intervallet $[a, b]$ mindre — om man till exempel gör det så litet att $|a - b| < 0.0005$ så har man hittat nollstället med tre decimalers noggrannhet.

Man gör intervallet hälften så stort genom att räkna ut mittpunkten m i intervallet $[a, b]$ och beräkna funktionsvärdet $f(m)$. Om nu $f(m)$ har samma tecken som $f(a)$ kan intervallet begränsas till $[m, b]$ (dvs man sätter a till m), annars till $[a, m]$ (dvs man sätter b till m). Detta upprepas sedan med det nya kortare intervallet, tills man har uppnått önskad noggrannhet.

I följande figur åskådliggörs några steg i halveringen:



Skriv en Matlabfunktion med anropet `zero = inthalv(a,b,tol)` som finner ett nollställe `zero` i intervallet `[a,b]` till en funktion med namnet `f`. Nollstället ska beräknas med noggrannheten `tol`. Du kan förutsätta att `f(a)` och `f(b)` har olika tecken. I Java skulle funktionen kunna se ut så här:

```
public double inthalv(double a, double b, double tol) {
    while (Math.abs(a - b) >= tol) {
        double m = (a + b) / 2;
        if (f(a) * f(m) > 0) {
            a = m;
        }
        else {
            b = m;
        }
    }
    return a;
}
```

Att testa om två tal `x` och `y` har samma tecken med `x * y > 0` är betydligt enklare än att skriva `x > 0 && y > 0 || x < 0 && y < 0`.

Kontrollfrågor

K1. Vad är skillnaden mellan Matlab och Maple?
K2. Vilken noggrannhet har man vid beräkningar i dubbel precision?
K3. Matlab har, precis som miniräknare, begränsad räknenoggrannhet. Hur kan detta yttra sig?
K4. Hur stort kan ett dubbel precisions-tal vara? Hur litet?
K5. Hur skriver man för att få hjälp om ett kommando i Matlab?
K6. Hur undviker man i Matlab att resultatet från en beräkning skrivs ut?
K7. Ge exempel på några inbyggda funktioner i Matlab.
K8. Vad skrivs ut när följande Matlabkommandon exekveras?

```
x = [1 4 9 16];
sqrt(x)
```

K9. Hur skriver man i Matlab för att plotta funktionen $f(x) = x^2$ i intervallet $[-2, 2]$?
K10. Hur skriver man i Matlab för att skapa en vektor `v` med de 100 elementen -1, 1, 3, 5, ..., 195, 197?
K11. Vad är en `m`-fil?
K12. Hur tar man reda på det första elementet i en vektor? Det sista?

Datorarbete

Matlab som miniräknare

Matlab fungerar bra som miniräknare. Alla vanliga matematiska operationer och funktioner finns inbyggda. Man kan spara resultat i variabler och använda dem senare.

D1. Starta Matlab genom att ge kommandot `matlab` i ett kommandofönster. Du får ett nytt fönster där du kan skriva Matlab-kommandon. Om du skriver `matlab -nojvm` skapas inte något nytt fönster; då får du skriva Matlab-kommandona i kommandofönstret. (I vissa versioner av Matlab kan det inträffa att hakparenteser `[` och `]` inte fungerar om man inte anger `-nojvm`.) Alternativt kan du använda `octave`.

D2. Prova Matlab som miniräknare: beräkna `3.5+2*4.5`, `pi/3`, `sin(pi/3)`, ...

D3. Prova variabler: sätt `x = pi/3`, beräkna `y = sin(x)`, `1/y`, ...

Vektorer

En vektor är en följd av värden som är numrerade 1, 2, 3, ... (observera att man börjar numrera från 1, inte från 0 som man gör i Java). Man kommer åt ett element genom att skriva vektorns namn och index inom vanliga parenteser `()` (inte hakparenteser `[]` som i Java). Exempel:

```
>> x = [1 3 4 5 10]
x = 1 3 4 5 10

>> x(2)
ans = 3

>> x(1) + x(length(x))
ans = 11

>> x(2) = x(2) + 1
x = 1 4 4 5 10
```

Funktioner kan ha vektorer som parametrar. Då appliceras funktionen på alla elementen:

```
>> sqrt(x)
ans = 1.0000 2.0000 2.0000 2.2361 3.1623
```

Man kan enkelt bilda ”regelbundna” vektorer genom att ange startvärde, steg och slutvärde:

```
>> x = 0 : 0.5 : 7
x = 0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
```

Om man har en vektor med x-värden och en vektor med funktionsvärden kan man rita upp (”plotta”) funktionen:

```
>> f = sin(x);
>> plot(x, f)
```

Man kan spara bilder som man ritat i Matlab (`Save`-kommandot i `File`-menyn i ritfönstret). Bilder kan sparas i många olika format. Om man vill inkludera en bild i ett LaTeX-dokument ska den vara i *pdf*-format (eller *jpg* eller *png*). Om man har sparat en bild i ”fel” format är det enkelt att konvertera till ett annat bildformat med ImageMagick-programmet `convert`. Exempel:

```
convert myfig.fig myfig.pdf
```

D4. Rita upp sinuskurvan enligt exemplet. Kurvan blir ojämn — rita en jämnare kurva.

D5. Rita en cosinuskurva i samma intervall i *samma* fönster. Kurvan ska ritas med röd färg (`help plot`).

D6. Rita i ett nytt fönster en kurva där x-värdena är punkterna i cosinusvektorn och y-värdena är punkterna i sinusvektorn. Ge kommandot `axis('equal')` för att få samma skala på båda axlarna. Förklara resultatet.

Matriser och ekvationssystem

I Javakursen kommer vi att arbeta med matriser, som är rektangulära ”rutnät”. Exempel på en matris med 3 rader och 3 kolonner där elementen är heltal:

$$a = \begin{pmatrix} 3 & 5 & 1 \\ 1 & 3 & 4 \\ 0 & 2 & 1 \end{pmatrix}$$

Matlab är väldigt bra på att hantera matriser. För att skapa en matris skriver man ungefär som när man skapar en vektor, men man skiljer på raderna med semikolon. Man kommer åt elementen genom att ge matrisens namn och radindex och kolonnindex, till exempel `a(2,3)` (i Java skulle man skrivit `a[1][2]`). Exempel:

```
>> a = [3 5 1; 1 3 4; 0 2 1]
a = 3 5 1
    1 3 4
    0 2 1
>> a(2,3)
ans = 4
```

I matematiken används matriser väldigt mycket. Du kommer för första gången att träffa på dem i kursen i Linjär algebra. Vi visar här bara ett exempel på hur man använder matriser (och vektorer) för att lösa ett linjärt ekvationssystem.

Ett linjärt ekvationssystem med tre ekvationer och tre obekanta kan se ut så här:

$$\begin{aligned} 3x + 5y + z &= 17 \\ x + 3y + 4z &= 23 \\ 2y + z &= 8 \end{aligned}$$

Koefficienterna framför x, y och z stoppar man in i en matris (det har vi redan gjort ovan). Talen i högerledet stoppar man in i en vektor där talen ligger ”ovanför” varandra, inte bredvid varandra. En sådan vektor kallas en kolonnvektor.

```
>> b = [17; 23; 8]
b = 17
    23
     8
```

Nu kan man lösa ekvationssystemet genom att skriva:

```
>> a \ b
ans = 1
      2
      4
```

D7. I hemuppgift H2 löste du ett ekvationssystem för hand. Lös samma ekvationssystem i Matlab.

D8. Försök lösa följande ekvationssystem:

$$\begin{aligned} 2x - y &= 3 \\ -4x + 2y &= 5 \end{aligned}$$

Förklara resultatet! Tips: tänk geometriskt ...

Ekvationslösning, egna funktioner

Det finns i Matlab en inbyggd funktion `fzero` som beräknar ett nollställe till en ekvation $f(x) = 0$. Som parametrar ger man funktionens namn och en punkt som ligger i närheten av nollstället, till exempel så här:

```
>> fzero('sin', 3)
ans = 3.1416
```

Om man ska hitta nollställen till en mera komplicerad funktion så måste man skriva en egen Matlab-funktion som beräknar funktionsvärden. Nedanstående Matlabfunktion beräknar värdet av funktionen $e^{-x} - e^{-2x} + 0.05x - 0.25$ i punkten `x`:

```
function fval = f(x)
% FVAL = F(X), compute the value of a function in x
fval = exp(-x) - exp(-2 * x) + 0.05 * x - 0.25;
```

Som du ser så skriver man inte riktigt likadant som i Java. Funktionsrubriken ser annorlunda ut, exponentialfunktionen heter `exp` i stället för `Math.exp` och man returnerar resultatet genom att tilldela "utparametern" `fval` ett värde i stället för att göra `return`.

D9. Matlabfunktionen `f` finns i filen */usr/local/cs/dod/lab5/f.m*. Kopiera filen till din egen katalog. Testa kommandona `help f` och `type f`.

D10. Funktionen har tre nollställen, samtliga större än 0. Använd `fzero` för att hitta nollställena. Plotta först funktionen så att du får en uppfattning om var nollställena ligger.

D11. Skriv in funktionen `inthalv` som du skrev i hemuppgift H3 i en fil med namnet *inthalv.m*. Använd `inthalv` för att beräkna de tre nollställena till funktionen `f` med fyra decimalers noggrannhet.

D12. Lös följande uppgift, om du har tid: Funktionen `inthalv` är inte särskilt användbar eftersom den är specialskriven för att hitta nollställen till en funktion med namnet `f`. Det vore betydligt mera praktiskt med en intervallhalveringsfunktion som kunde anropas med namnet på en funktion som parameter, till exempel med `zero = inthalv('f',a,b,tol)` eller `zero = inthalv('sin',a,b,tol)`, på det sätt som man anropar `fzero`.

Skriv om `inthalv`-funktionen så att den fungerar på detta sätt och testa att den fungerar. När du ska räkna ut ett funktionsvärde inuti `inthalv` måste du utnyttja Matlabfunktionen `feval`; studera beskrivningen av denna funktion i hjälptexten.

Datorer och datoranvändning, godkända laborationsuppgifter

Skriv ditt namn och din namnteckning nedan:

Namn: .....

Namnteckning: .....

Godkänd laboration	Datum	Laborationsledarens namnteckning
1		
2		
3		
4		
5		