NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Løsningsforslag: Eksamensoppgave i TDT4110 Informasjonsteknologi - grunnkurs

Eksamensdato:	**2014-12-06**
Eksamenstid (fra-til):	**09:00 – 13:00**
Hjelpemiddelkode/Tillatte hjelpemidler:	**Godkjent kalkulator**

Annen informasjon:

Oppgavesettet inneholder 4 oppgaver. Det er angitt i prosent hvor mye hver oppgave og hver deloppgave teller ved sensur. Les igjennom hele oppgavesettet før du begynner å løse oppgavene. Disponer tiden godt! Gjør rimelige antagelser der du mener oppgaveteksten er ufullstendig, skriv kort hva du antar.

Svar kort og klart, og skriv tydelig. Er svaret uklart eller lenger enn nødvendig trekker dette ned.

Målform/språk:	**Bokmål**

Oppgave 1: Flervalgsoppgave (25%)

Bruk de to vedlagte svarskjemaene for å svare på denne oppgaven (ta vare på det ene selv). Du kan få nytt ark av eksamensvaktene dersom du trenger dette. Kun ett svar er helt riktig. For hvert spørsmål gir korrekt avkryssing 1 poeng. Feil avkryssing eller mer enn ett kryss gir −1/2 poeng. Blankt svar gir 0 poeng. Du får ikke mindre enn 0 poeng totalt på denne oppgaven. Der det er spesielle uttrykk står den engelske oversettelsen i parentes.

1. Hvilken fundamental aktivitet innen programvareutviklingsprosessen fokuserer på å endre programvaren for å møte endrede kunde- og markedskrav?
    a. Programvarespesifikasjon
    b. Programvareutvikling
    c. Programvarevalidering
    d. Programvareevolusjon

2. Hva er et analogt signal?
    a. Et kontinuerlig signal hvor den variable egenskap er gitt av en diskret funksjon, som gir verdier fra et definert og begrenset område.
    b. Et kontinuerlig signal hvor en variabel egenskap (f.eks. amplitude eller frekvens) representerer informasjonen som overføres.
    c. Et diskret signal som representeres ved hjelp av nuller og enere.
    d. En kombinasjon av alternativ a og b.

3. Hvilken type løkkestruktur er garantert å utføre handlingen minst en gang?
    a. pre-test løkke (pretest loop).
    b. post-test løkke (posttest loop).
    c. begge typer.
    d. ingen av typene.

4. Omtrent hvor mange ganger raskere er en 1 GHz - prosessor i forhold til en på 2 MHz.
    a. Halvparten så rask.
    b. Like rask.
    c. Dobbelt så rask.
    d. 500 ganger så rask.

5. Hva sier Nyquist-regelen?
    a. at samplingsrate ved lyd må være minst det dobbelte i forhold til høyeste frekvensen.
    b. at lyd over 20000Hz ikke kan høres av det menneskelige øret.
    c. at tapsfri komprimering ikke er mulig for lyd.
    d. at lyddata tapsfritt kan komprimeres med maksimalt en faktor 2*pi.

6. Hvilken programvareprosessmodell bør velges for et prosjekt der det skal utvikles et helt nytt system hvor eksisterende komponenter ikke finnes og kunden er usikker på hvordan systemet skal være?
    a. Vannfallsmodellen.
    b. Inkrementell utvikling.
    c. Gjenbruksorientert systemutvikling.
    d. Havmodellen.

7. Hva er hensikten med et paritetsbit i digitale signaler?
    a. Forteller hvor meldingen skal sendes.
    b. Gjør meldingene raskere å overføre (komprimering).
    c. Bidrar til å detektere feil i digitale signaler.
    d. Krypterer signaler så overføringen av data blir sikrere.
8. Kompleksiteten til sortering ved innsetting (insertion sort) er
    a. $\Theta(n)$.
    b. $\Theta(n \log n)$.
    c. $\Theta(n^2)$.
    d. $\Theta(2n)$.
9. En moderne prosessor er typisk bygd opp av mange millioner små…
    a. Dioder.
    b. Magneter.
    c. Transistorer.
    d. Kondensatorer.
10. En byte med minne i datamaskinen kan lagre hvor mye?
    a. 16 bits.
    b. 8 flyttall.
    c. fire ASCII-tegn.
    d. en heltallsverdi mellom 0 og 255.
11. Hvilken av de følgende er en kjent fordel med vannfallsmodellen?
    a. Tar hensyn til brukerkrav som endrer seg i løpet av prosjektet.
    b. Gjør prosessen synlig og enklere å monitorere for prosjektlederen.
    c. Får tidlige versjoner av systemet raskt ut til kunden.
    d. Åpner for kontinuerlig tilbakemelding fra brukerne av systemet.
12. Morsekode representerer bokstaver som sekvenser av prikk og strek som er
    a. like lange for alle bokstaver i alfabetet.
    b. kortere for bokstaver tidlig i alfabetet, lenger for bokstaver sist i alfabetet.
    c. kortere for vokaler, lenger for konsonanter.
    d. kortere for bokstaver som forekommer hyppig i vanlig tekst, lenger for sjeldnere bokstaver.
13. Hvilken av disse er en korrekt gjengivelse av teoribokas definisjon av en algoritme? "En algoritme er et ordnet sett av…"
    a. "… entydige, utførbare skritt som definerer en terminerende prosess" (unambiguous, executable steps that defines a terminating process).
    b. "… entydige, effektive skritt som definerer en utførbar prosess" (unambiguous, efficient steps that defines an executable process).
    c. "… velformede, effektive skritt som definerer en terminerende prosess" (well-formed, efficient steps that defines a terminating process).
    d. "…velformede, utførbare skritt som definerer en effektiv prosess" (well-formed, efficient steps that defines an efficient process).
14. En datamaskin går i en uendelig løkke som kalles
    a. Det naturlige kretsløpet.
    b. Hent-Utfør kretsløpet.
    c. Det evige kretsløpet.
    d. Beregnings-kretsløpet.

15. Hva er korrekt binær representasjon av 'NTNU' i 8 bits ASCII?
    a. `01001110 01010100 01001110 01010101.`
    b. `01100001 01100100 01110011 01100110.`
    c. `01101110 01110101 01110100 01001110.`
    d. `01100010 01010101 01010010 01010000.`

16. I hvilket tilfelle er det mest nyttig å bruke gjenbruksorientert systemutvikling?
    a. Når det finnes tilgjengelig programvare som kan gjøre oppgaver systemet skal utføre.
    b. Når det skal lages programvare for å håndtere resirkulering av søppel eller lignende systemer.
    c. Når det skal gjenbrukes ideer fra tidligere prosjekter.
    d. Når det skal gjenbrukes systemutviklere og systemdesignere fra tidligere prosjekter.

17. Hva står forkortelsen ISP for?
    a. Internet Service Provider.
    b. Information Security Protocol.
    c. Internet Security Protocol.
    d. Information Super Pool.

18. Kompleksiteten til binærsøk er
    a. Θ(n) hvis lista er sortert og Θ(n log n) hvis den er usortert.
    b. Θ(log n) hvis lista er sortert og Θ(2 log n) hvis lista er usortert.
    c. Θ(log n) hvis lista er sortert og Θ(n) hvis lista er usortert.
    d. Θ(log n) hvis lista er sortert. Binærsøk er ubrukelig hvis lista er usortert.

19. RAM
    a. Husker alle verdiene når strømmen kuttes.
    b. Er alltid inndelt i blokker på 1 kilobyte.
    c. Betyr Random Access Memory.
    d. Kan trygt fjernes uten at maskinen slutter å fungere.

20. Hva står bokstavene i RGB for?
    a. Red, Green, Blue.
    b. Readable Graphics Byte.
    c. Raster Grayscale Balance.
    d. Realtime GPU Backlog.

21. Hva kalles aktiviteten som har fokus på å identifisere den overordnede strukturen for et system inkludert dets sub-systemer?
    a. Hoveddesign.
    b. Arkitekturdesign.
    c. Interfacedesign.
    d. Komponentdesign.

22. MODEM er en forkortelse for
    a. MOdulator / DEModulator.
    b. Massive Online Digital Electric Messaging.
    c. MOnitored Data EMission.
    d. Mapping Of Digital Electronic Mail.

23. ASCII-kode representerer bokstavene A til Z som sekvenser av 0 og 1 som er
    a. like lange for disse bokstavene i alfabetet.
    b. kortere for bokstaver tidlig i alfabetet, lenger for bokstaver sist i alfabetet.
    c. kortere for vokaler, lenger for konsonanter.
    d. kortere for bokstaver som forekommer hyppig i vanlig tekst, lenger for sjeldnere bokstaver.

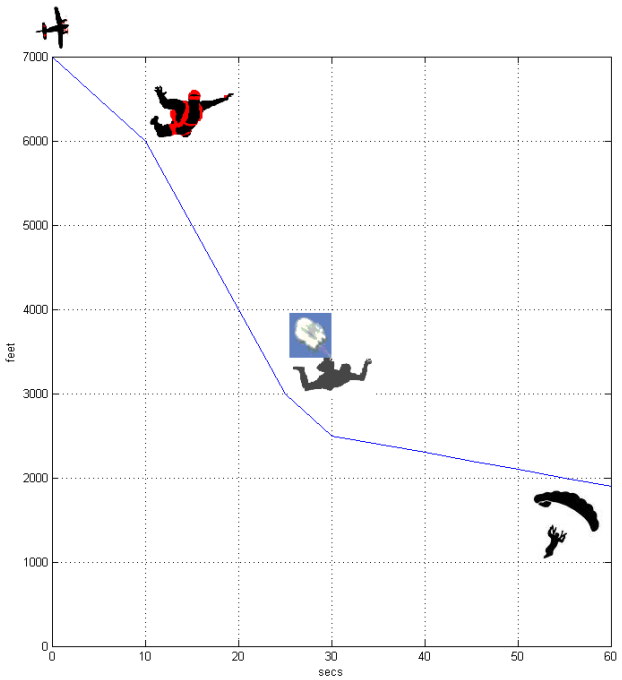
24. Et nettverk som knytter sammen datamaskiner og utstyr i et begrenset område som et kontor, bygning eller i en bolig betegnes med forkortelsen:
    a. LAN.
    b. MAN.
    c. PAN.
    d. WAN.

25. VPN (Virtual Private Network) kan gi mottageren inntrykk av at en reisende ansatt sin bærbare PC befinner seg innenfor bedriftens nettverk ved at meldinger fra denne PC'en
    a. plasseres inni en kryptert datapakke for ekstern oversendelse.
    b. sendes med en tidsforsinkelse.
    c. sendes ekstra hurtig, med høy prioritet.
    d. sendes med en falsk avsenderadresse som inneholder et virus.

Oppgave 2 Programmering Fallskjerm (25%)

(I denne oppgaven kan det være gunstig å kalle funksjoner som du har laget i tidligere deloppgaver. Selv om du ikke har fått til den tidligere oppgaven, kan du kalle funksjon derfra med antagelse om at den virker som spesifisert i oppgaveteksten.)

NTNU fallskjermklubb (NTNU-FSK) trenger hjelp til å lage et nytt opplærings- og administrasjonsprogram. De har bedt firmaet ditt (ITGK) om hjelp, og du har fått jobben med å programmere funksjoner som beskrevet i deloppgavene under.
Vi benytter en litt forenklet versjon av jordens fysiske lover:



Figur 1. Hopp fra 7000 fot.

En fallskjermhopper faller (med konstant/gjennomsnittlig hastighet) 100 fot pr. sekund de 10 første sekundene, og deretter med konstant topphastighet på 200 fot pr. sekund til skjermen må åpnes i 3000 fots høyde (se figur 1). Hvis man mot normalt hopper ut under 3000 fot må skjermen utløses umiddelbart (etter 0 sekunder).

Medlemsdatabasen til NTNU-FSK ligger lagret på en fil 'members.txt', med følgende format:

NAVN;ID;VEKT;SKJERMST.
Eksempel på innholdet i filen:

```
Frank Stank;D-49334;75;120
Bjarne Stor;C-49335;95;150
Dumbo Ear;D-50105;450;750
 Peter Pan;A-12345;30;100
```

Oppgave 2a (5%)

Lag en funksjon `inputPerson` som leser inn navn, id, vekt og skjermstørrelse fra tastaturet, og returnerer en liste med verdier for navn, id, vekt og skjermstørrelse. Merk at navn og id er tekststrenger, mens vekt og skjermstørrelse er heltall. Du kan anta at bruker skriver inn lovlige verdier og det trengs ikke unntakshåndtering.
Eksempel på kjøring (tekst med understreking skrives inn av brukeren):

```
>>> person = inputPerson()
Name: Fredrik Olsen
ID: B-77777
KG: 80
Size: 240
>>> person
['Fredrik Olsen', 'B-77777', 80, 240]
>>>
```

Mulig løsning 2a:

```
def inputPerson():
  name=input('Name: ')
  ID=input('ID: ')
  KG=int(input('KG: '))
  size=int(input('Size: '))
  return[name,ID,KG,size]
```

Oppgave 2b (5%)
Lag en funksjon `readDbFile` som leser inn hele medlemsbasen til en tabell (liste av lister) med følgende elementer: `name`, `id`, `weight` og `size` (se beskrivelse ovenfor). `name` og `id` er tekststrenger, mens `weight` og `size` er heltall. Du kan anta at filen finnes, at det ikke oppstår noen problemer ved åpning/lukking, og at filen ikke inneholder noen blanke eller ugyldige linjer.
Funksjonen skal ha inn-parameter `filename` og returnere tabellen (liste av lister).
Eksempel på kjøring:

```
>>> db=readDbFile('members.txt')
>>> for line in db:
      print(line)

['Frank Stank', 'D-49334', 75, 120]
['Bjarne Stor', 'C-49335', 95, 150]
['Dumbo Ear', 'D-50105', 450, 750]
['Peter Pan', 'A-12345', 30, 100]
>>>
```

Mulig løsning 2b:

```
def readDbFile(filename):
  f = open(filename,'r')
  db=[] # Create empty list for the results
  for line in f:
    line = line.strip() # Remove space and white spaces
    slist = line.split(';')
    info=[slist[0],slist[1],int(slist[2]),int(slist[3])]
    db.append(info)
  f.close()
  return db
```

Oppgave 2c (5%)
Lag en funksjon `printMemberList` som skriver ut innholdet av tabellen `db` (listen av lister som beskrevet ovenfor) på skjermen med overskrifter og format som vist i eksemplet på kjøring (der en antar at `db` har samme innhold som i 2b):

```
>>> printMemberList(db)
NAVN            ID-NR   VEKT kg. SKJERMSTØRRELSE
Frank Stank     D-49334   75 kg  120 kvadratfot
Bjarne Stor     C-49335   95 kg  150 kvadratfot
Dumbo Ear       D-50105  450 kg  750 kvadratfot
Peter Pan       A-12345   30 kg  100 kvadratfot
>>>
```

Det skal settes av 15 tegn til NAVN, 9 tegn til ID-NR, 5 tegn/siffer til VEKT, og 4 tegn/siffer til SKJERMSTØRRELSE. Du kan anta at databasen ikke har innhold som går utover de avsatte antall tegn for hvert felt. Om verdier krever mindre plass en tallet på avsatte tegn, skal resten av feltbredda i skjermutskrifta fylles av blanke. Første felt skal være venstrejustert, mens de påfølgende skal være høyrejustert.

Funksjonen skal ha inn-parameter `db`, ingen retur-verdi.

Mulig løsning 2c:

```
def printMemberList(db):
  print('NAVN            ID-NR   VEKT kg. SKJERMSTØRRELSE')
  for line in db:
    s=line[0].ljust(15)
    s+=line[1].rjust(9)
    s+=str(line[2]).rjust(5)+' kg '
    s+=str(line[3]).rjust(4)+' kvadratfot'
    print(s)
```

Oppgave 2d (5%)
Lag en funksjon `addPerson` med inn-parameteren `filename` (filnavnet til databasen). Funksjonen skal be bruker skrive inn informasjon om en person beskrevet ved navn, id, vekt og skjermstørrelse og lagre dette i variabelen `person`. Funksjonen skal så lese inn databasen som ligger lagret i filen `filename` til datastrukturen `db` som er en tabell (liste av lister) som beskrevet i oppgave 2b. Deretter skal opplysningene om den nye personen legges til i `db` og i filen `filename`. Bruk riktig format: se «Eksempel på innholdet i filen» ved figur 1 (se side 6). Returverdi `db` skal inneholde den oppdaterte databasen.
Du trenger ikke å skrive kode for feilhåndtering i forbindelse med å lese eller skrive fra/til fil.
Eksempel på kjøring (alt som er understreket er informasjon skrevet inn av bruker, og alt som er i **fet skrift** er kjøring av funksjonen og endringen av resultatet ved kjøring):

```
>>> db=addPerson('members.txt')
Name: Santa Klaus
ID: H-12345
KG: 155
Size: 380
>>>
>>> printMemberList(db)
NAVN            ID-NR   VEKT kg. SKJERMSTØRRELSE
Frank Stank     D-49334   75 kg  120 kvadratfot
Bjarne Stor     C-49335   95 kg  150 kvadratfot
Dumbo Ear       D-50105  450 kg  750 kvadratfot
Peter Pan       A-12345   30 kg  100 kvadratfot
Santa Klaus     H-12345  155 kg  380 kvadratfot
>>>
```

Mulig løsning 2d:

```
def addPerson(filename):
  person = inputPerson()
  db = readDbFile(filename)
  db.append(person)
  f = open(filename,'a')
  s = person[0]+';'+person[1]+';'+str(person[2])+';'+str(person[3])+'\n'
  f.write(s)
  f.close()

  return db
```

Oppgave 2e (5%)
For en fallskjermhopper er det veldig viktig å være klar over hvor mange sekunder man kan vente før man må åpne fallskjermen (se figur 1). Lag en funksjon `feet2seconds` som regner ut hvor mange sekunder det tar å falle fra et oppgitt antall fot ned til 3000 fot (inn-parameter `feet`, og retur-verdi `seconds`). Bruk informasjon gitt i starten av oppgave 2 (forklaringen til figur 1) til å beregne riktig tid. Hvis antall fot er under 3000 skal funksjonen returnere 0.
Eksempler på bruk:

```
>>> feet2seconds(12500)
52.5
>>> feet2seconds(7000)
25.0
>>> feet2seconds(2000)
0
>>>
```

Mulig løsning 2e:

```
def feet2seconds(feet):
    if feet > 4000:
        return 10 + (feet-4000)/200
    elif feet > 3000:
        return (feet-3000)/100
    else:
        return 0
```

Oppgave 3 Programmering Værstasjon (30%)

Du skal behandle data fra en værstasjon for et større antall dager. Dataene ligger lagret som flyttall i en tabell (liste av lister) som heter `weatherData`. Hver rekke (hver liste i lista) representerer måledata for *en dag* og har tre elementer av ulike typer måledata. De tre typene med måledata er maksimumstemperatur, minimumstemperatur og nedbørsmengde. Vi refererer til dagene slik at dataene i første rekke (`weatherData[0]`) er for dag nr 1, dataene for andre rekke (`weatherData[1]`) er for dag nr 2 osv.
Eksempel på utskrift av rekker i `weatherData` kan være:

```
>>> for row in weatherData:
      print(row)

[12.0, 2.4, 8.2]
[6.1, 0.6, 11.9]
[8.3, -3.5, 0.0]
[11.6, -5.2, 0.0]
[15.3, 2.8, 14.3]

>>> weatherData[0]
[12.0, 2.4, 8.2]
>>> weatherData[1]
[6.1, 0.6, 11.9]
>>>
```

Oppgave 3a (10%)
Skriv funksjonen `weatherStats` som tar inn `weatherData` som parameter. Funksjonen skal gå igjennom dataene og utfra det skrive et sammendrag som vist under. Det vil si at den skal skrive ut antall dager i perioden, totalmengden av nedbør i hele perioden, samt at den skal liste den aller laveste og aller høyeste temperaturen sammen med *nummeret* på dagene for disse temperaturene. Eksempel på kjøring for dataene vist i den grå boksen ovenfor:

```
>>> weatherStats(weatherData)
There are 5 days in the period.
The highest temperature was 15.3 C on day number 5
The lowest temperature was -5.2 C on day number 4
There was a total of 34.4 mm rain in the period
>>>
```

Mulig løsning 3a:

```
def weatherStats(weatherData):# Problem 3a
  days=len(weatherData)
  htemp= weatherData [0][0]
  ltemp= weatherData [0][1]
  hday=lday=1
  rain=0
  daycount=0
  # Find lowest and highest temp and date
  for daydata in weatherData:
    rain=rain+daydata[2]
    daycount+=1
    if daydata[0]>htemp:
      hday=daycount
      htemp=daydata[0]
    if daydata[1]<ltemp:
      lday=daycount
      ltemp=daydata[1]

  print('There are',days,'days in the period.')
  print('The highest temperature was',htemp,'C on day number',hday)
  print('The lowest temperature was',ltemp,'C on day number',lday)
  print('There was a total of',rain,'mm rain in the period')
```

Oppgave 3b (10%)
Skriv en funksjon `coldestThreeDays` som tar inn parameteren `weatherData` (som definert over). Funksjonen skal finne den perioden av tre sammenhengende dager som hadde den laveste gjennomsnittlige minimumstemperaturen. Den skal returnere nummeret på første dagen i denne tredagersperioden. Dersom det er flere perioder som er like kalde, så skal den returnere kun den siste av disse periodene. Et eksempel på en kjøring av denne funksjonen for `weatherData` som definert tidligere i oppgaven gir:

```
>>> coldestThreeDays(weatherData)
2
>>>
```

Mulig løsning 3b:

```
def coldestThreeDays(weatherData): # Problem 3b
  ctemp=(weatherData[0][1]+weatherData[1][1]+weatherData[2][1])/3
  cday=1
  for i in range(1,len(weatherData)-2):
    averagetemp=(weatherData[i][1]+weatherData[i+1][1]+weatherData[i+2][1])/3
    if averagetemp<=ctemp:
      cday=i+1
      ctemp=averagetemp
  return cday
```

Oppgave 3c (10%)
Værstasjonen har nettopp rapportert data for ytterligere en dag. Den kommer som en tekststreng lagret i variabelen `extraData` som har formatet som vist under:

```
>>> extraData
'max=23.5, min=9.3, 5.1mm'
>>>
```

Skriv en funksjon `addNewDay` som tar `extraData` samt `weatherData` som parametere, og som returnerer en ny versjon av `weatherData` som er oppdatert med de nye dataene på slutten av tabellen.

Under er vist et eksempel på kjøring (selve kjøringen av funksjonen `addNewDay` og endringer fra funksjonen er vist i **fet skrift**):

```
>>> for row in weatherData:
      print(row)

[12.0, 2.4, 8.2]
[6.1, 0.6, 11.9]
[8.3, -3.5, 0.0]
[11.6, -5.2, 0.0]
[15.3, 2.8, 14.3]
>>>
>>> extraData
'max=23.5, min=9.3, 5.1mm'
>>>
>>> weatherData = addNewDay(extraData,weatherData)
>>> for row in weatherData:
      print(row)

[12.0, 2.4, 8.2]
[6.1, 0.6, 11.9]
[8.3, -3.5, 0.0]
[11.6, -5.2, 0.0]
[15.3, 2.8, 14.3]
[23.5, 9.3, 5.1]
>>>
```

Mulig Løsning 3c:

```
def addNewDay(string,weatherData): # Problem 3c
  itemlist=string.split()
  mtemp=float(itemlist[0][4:-1])
  ltemp=float(itemlist[1][4:-1])
  rain=float(itemlist[2][:-2])
  weatherData.append([mtemp,ltemp,rain])
  return weatherData
```

Oppgave 4 Kodeforståelse (20%)

Oppgave 4a (5%)
Hva returneres ved kjøring av funksjonen `myst([1,2,3,3,2,1])` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst()` gjør? (2 %)

```
def myst(A):
  L=len(A)
  if (L>1):
    B=A[0]*A[L-1]
    return B+myst(A[1:L-1])
  return 0
```

Løsning 4a:

14
Forklaring: Går fra ytterst mot midten og multipliserer to og to ledd, og legger sammen resultatet (1*1+2*2+3*3).

Oppgave 4b (5%)
Hva blir skrevet ut når man kjører koden nedenfor? (3 %)
Forklar med en setning hva funksjonen `myst_b( )` gjør? (2 %)

```
def myst_b(W):
  # create a 2d-list with W x W zeros
  table = [[0 for i in range(W)] for j in range(W)]
  for a in range(W):
    table[a][a]=1
    b=0
    while a-b > 0 and a+b < W-1:
      b+=1
      table[a-b][a+b]=1
      table[a+b][a-b]=1
  return table

for line in myst_b(4):
  print(line)
```

Løsning 4b:

```
[1, 0, 1, 0]
[0, 1, 0, 1]
[1, 0, 1, 0]
[0, 1, 0, 1]
```

Forklaring: Lager en tabell med W rader og W kolonner, hvor innholdet blir et "sjakkrutemønster" av 0 og 1.

Oppgave 4c (5%)
Hva returneres ved kjøring av funksjonen `myst_c('RBHOOASDUEØGNGEBLSOIURMNGTD')` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst_c()` gjør? (2 %)

```
def myst_c(A):
  B=''
  for x in range(0,len(A),3):
    B=B+A[x]
  return B
```

Løsning 4c:

```
'ROSENBORG'
```

Forklaring: Returnerer en streng med hvert tredje tegn i parameteren A.

Oppgave 4d (5%)
I et program som skal trene ungdomsskoleelever i matematikk, trenger vi en funksjon for å sjekke korrekt nøsting av parenteser i uttrykk der tre ulike parentestyper er tillatt. Funksjonen trenger ikke å sjekke at uttrykket ellers er fornuftig, kun at parenteser kommer i lovlig rekkefølge og går opp mhp antall og plassering av alle start- og sluttparenteser. Vi har også skrevet tre print-setninger som kaller funksjonen for å teste om den virker. Koden er vist her (linjenummer til venstre ikke del av koden men er tatt med så du lettere kan vise til spesifikke kodelinjer i svaret ditt):

```
1     PAREN=['(','[','{',')',']','}']
2
3     def test(expression):
4       parentheses_list = []
5       for char in expression:
6         if char in PAREN[:3]: # start-parenthesis found
7           # put corresponding end-parenthesis in the back of the list
8           parentheses_list.append(PAREN[PAREN.index(char)+3])
9         elif char in PAREN[3:]: # end-parenthesis found
10          if char not in parentheses_list: # not matched start-parenthesis
11            return False
12          else:
13            parentheses_list.remove(char)
14      return parentheses_list
15
16    def main():
17      print('A:', test('{a+4*[b-2*(c+5)]/11}'))   #should give True
18      print('B:', test('{a+4*[b-2*(c+5])/11}'))   #should give False
19      print('C:', test('{a+4*[b-2*(c+5])/11}}'))  #should give False
```

Som kommentarene sier skulle utskriften fra de tre print-setningene ha blitt hhv. True, False og False. Når vi kjører `main`-funksjonen får vi imidlertid:

```
>>> main()
A: []
B: []
C: False
>>>
```

Det er to feil i koden:

(1) funksjonen returnerer tom liste (`[]`) i stedet for en boolsk verdi for både A og B. Hvis denne feilen blir rettet, vil kjøring av programmet indikere den andre feilen:

```
>>> main()
A: True
B: True
C: False
>>>
```

(2) som man ser over: funksjonen returnerer True for noen uttrykk som skulle gitt False, som for linje B her.

Spørsmål: Forklar hvilke kodelinjer som forårsaker feil (1) og feil (2) og hvordan de enklest kan rettes. I begge tilfeller skal det være mulig å rette feilen bare ved å endre noe i eksisterende kodelinjer, det skal ikke være nødvendig å legge til nye kodelinjer.

Løsning 4d:

Må endre linje 10, 13 og 14:

```
10          if char != parentheses_list[-1]:    # FIXED
11            return False
12          else:
13            parentheses_list.pop()            # FIXED
14      return parentheses_list==[]             # FIXED
```

Forklaring:
Line 10 sjekket ikke rekkefølge. Endret til å sjekke med parentes som ble sist lagt til lista.
Linje 13 fjernet første forekomst av gitt parentes. Endret til å fjerne den som ble sist lagt til lista.
Linje 14 returnerte det som var igjen av liste. Endret til å sjekke om lista var tom!

Svarskjema flervalgsoppgave (sjablong – korrekte svar synlig)

Kandidatnummer: _____ Program: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgavenr	***A***	***B***	***C***	***D***
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				
1.21				
1.22				
1.23				
1.24				
1.25				

Svarskjema flervalgsoppgave (sjablong – feil svar synlig)

Kandidatnummer: _____ Program: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgavenr	***A***	***B***	***C***	***D***
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				
1.21				
1.22				
1.23				
1.24				
1.25				