NTNU
Det skapende universitet

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4110 Informasjonsteknologi – grunnkurs: Løsningsforslag

Eksamensdato: **11 / 8 - 2014**

Målform/språk: **Bokmål**

Innhold:

- Oppgave 1: Flervalgsoppgave (25 %)
- Oppgave 2: Grunnleggende programmering (20 %)
- Oppgave 3: Kodeforståelse (15 %)
- Oppgave 4: Mer programmering (40 %)

Oppgave 1: Flervalgsoppgave (25 %)

Bruk de to vedlagte svarskjemaene for å svare på denne oppgaven (ta vare på den ene selv). Du kan få et nytt ark av eksamensvaktene hvis du trenger det. **Kun et svar er helt riktig.** For hvert spørsmål gir korrekt avkryssing (kun et kryss) 1 poeng. Feil avkryssing, eller mer enn et kryss (gardering) gir -½ poeng. Blankt svar gir 0 poeng. Du får ikke mindre enn 0 poeng totalt på denne oppgaven.

1) Hvilken tjeneste/applikasjon på Internett krever vanligvis IKKE lav tidsforsinkelse?

a) On-demand Video.
b) Interaktiv audio og video.
c) Online gaming.
d) IP telefoni.

2) Hvilket alternativ er IKKE en del av fossefallsmetoden

a) Kravanalyse og definisjon
b) System- og programvaretesting
c) Brukergrensesnittvurdering
d) Integrasjon og systemtesting

3) For et problem av størrelsen n finnes fire algoritmer med forskjellig tids-kompleksitet. Hvilken vil bruke lengst tid (i verste fall) på store problemer.

a) O(1) (konstant tid)
b) O(n) (lineær tid)
c) $O(n^2)$ (kvadratisk tid)
d) $O(2^n)$ (eksponentiell tid)

4) I følge Nyquist-regelen er samplefrekvensen for lyd

a) halvparten av de frekvensene et menneske kan høre
b) den samme som de frekvensene et menneske kan høre
c) det dobbelte av de frekvensene et menneske kan høre
d) 3 ganger de frekvensene et menneske kan høre

5) Hva er en protokoll

a) En definisjon av hva datakommunikasjon er
b) Et sett med regler som gjør det mulig for to datamaskiner å kommunisere
c) Regler som bestemmer syntaksen i et programmeringsspråk
d) En definisjon av hvordan TCP/IP fungerer

6) Den riktige rekkefølgen fra minst til størst er

a) giga, kilo, mega, tera
b) kilo, mega, giga, tera
c) tera, kilo, mega giga
d) kilo, mega, tera, giga

7) Gitt følgende funksjon:

```
funksjon(n):
if n < 0:
 return n + funksjon(n+1)
else if n> 0:
 return n + funksjon(n-1)
else:
 return 0
```

Hva kalles den kodeblokken som kjøres hvis `n==0`?

a) rekursiv del (recursive case)
b) iterasjonsdel (iterative case)
c) basistilfelle (base case)
d) returklausul (return case)

8) ALU brukes i

a) Instruction Fetch
b) Instruction Execution
c) Result Return
d) Instruction Decode

9) Hvilken av disse forkortelsene er en kjent prosessmodell innen systemutvikling

a) SCRUP
b) UTML
c) RUP
d) RAM

10) En digital-til-analog konverterer (DAC)

a) Endrer digital informasjon til analog
b) Konverterer kontinuerlig lyd til digital lyd
c) Konverterer lyd til et elektrisk signal
d) Gir tilnærmede verdier

11) Hvilket av de følgende er ikke et høy-nivå språk

a) Java
b) C
c) Assembly
d) Visual Basic

12) Når er et sekvensielt søk effektivt?

a) Når datamengden er sortert
b) Når dataene er tall
c) Sekvensielle søk er aldri effektive
d) Når det det letes etter ligger tidlig i datamengden

13) Sangen «A little bit» er 3 minutter og 47 sekunder lang. Hvor mange bit trengs for å lagre den i stereo på en vanlig musikk-CD?

a) 1 411 200
b) 40 042 800
c) 84 672 000
d) 320 342 400

14) Hva er VPN

a) En metode som brukes for å oversette logiske navn til et IP-nummer
b) En del av TCP/IP spesifikasjonen
c) En måte å etablere en trygg/kryptert kommunikasjonskanal mellom to maskiner
d) En metode en internettleverandør bruker for å distribuere sensitivt innhold

15) En router er

a) en datamaskin som forbinder flere nettverk
b) et program som sender informasjonspakker mellom 2 datamaskiner
c) et program som setter sammen informasjonspakkene til meldinger før den leveres til mottaker
d) en datamaskin som er koblet på internett

16) Gitt navnelisten «Alice, Byron, Carol, Duane, Elaine, Floyd, Gene, Henry, Iris». Hvilken søkealgoritme vil finne Carol først (gjøre færrest sammenligninger)?

a) Binærsøk
b) Sekvensielt søk
c) Begge vil finne Carol like raskt
d) Svaret er avhengig av hvordan binærsøkalgoritmen er implementert

17): En byte hukommelse kan lagre

a) en av 1024 forskjellige tall
b) et ord
c) et ASCII tegn
d) en blokk

18) Hvor mange steg er det i Fetch/Execute syklusen

a) 3
b) 4
c) 5
d) 6

19) Programtelleren endres direkte av instruksjoner som kalles

a) Add and Multiply
b) Branch and Jump
c) Input and Output
d) Now and Next

20) Hvordan representeres det heksadesimale tallet A8 binært?

a) 10101000
b) 10010100
c) 11001000
d) 10001100

21) Forkortelsen DDOS i pensum står for…
   a) Digital Disk Operating System
   b) Double Density Optical Storage
   c) Distributed Denial Of Service
   d) Data Directory On Site

22) Hovedforskjellen mellom Boehm sin spiralmodell for programvareutvikling og andre, tidligere prosessmodeller var…
   a) eksplisitt fokus på analyse og håndtering av risiko
   b) eksplisitt fokus på arbeidsmiljøet i programvarebedriftene
   c) eksplisitt fokus på gradvis kompetanseutvikling i prosjektteamet
   d) eksplisitt fokus på gradvis kompetanseutvikling i bedriftsledelsen

23) Person A skal sende en konfidensiell melding til person B. Hva slags krypteringsnøkler skal i så fall brukes?
   a) A krypterer med A sin private nøkkel, B dekrypterer med A sin offentlige nøkkel
   b) A krypterer med B sin private nøkkel, B dekrypterer med A sin offentlige nøkkel
   c) A krypterer med B sin offentlige nøkkel, B dekrypterer med B sin private nøkkel
   d) A krypterer med B sin offentlige nøkkel, B dekrypterer med A sin offentlige nøkkel

24) Brannmur (firewall) er en type sikkerhetsteknologi. Hva er den mest korrekte og relevante påstanden når det gjelder brannmurer og trusler fra såkalte trojanske hester ('Trojan Horses')?
   a) Brannmurer kan verne mot trojanske hester ved å hindre uventet internett-trafikk fra utsida og inn til et system.
   b) Brannmurer kan verne mot trojanske hester ved å hindre uventet trafikk fra innsiden og ut til internett.
   c) Brannmurer kan verne mot trojanske hester ved å advare brukerne mot å åpne falske epost-meldinger.
   d) Brannmurer gir IKKE vern mot trojanske hester. Bare antivirusprogramvare er effektivt mot trojanske hester.

25) Hva er en viktig forskjell mellom systemtesting og akseptansetesting?
   a) Systemtesting fokuserer på å finne feil i et program, mens akseptansetesting fokuserer på de delene som fungerer.
   b) Systemtesting bruker gjerne oppkonstruerte testdata mens akseptansetesting bruker data fra kunden som skal ha systemet.
   c) Systemtesting tester bare systemet modul for modul, mens akseptansetesting tester hele systemet i et.
   d) Systemtesting kan gjøres av personell som ikke kan programmere, mens akseptansetesting må gjøres av personell som også kan programmere, for å rette uakseptable feil.

Oppgave 2: Grunnleggende programmering (20 %)

Spillet Sudoku handler om å fylle 9 rader, kolonner og kvadrater med alle tallene fra og med 1 til og med 9. Se figur over spillebrettet helt til høyre med tre eksempler på lovlig utfylt kolonne, rad og kvadrat. Eksempel på lovlig utfylt kolonne: →

8
9
1
2
3
4
5
6
7

Eksempel på lovlig utfylt rad:

4	5	6	1	2	3	9	8	7

Eksempel på lovlig utfylt kvadrat:

1	3	5
2	4	6
9	8	7

8								
9								
1								
2						1	3	5
3						2	4	6
4	5	6	1	2	3	9	8	7
5								
6								
7								

Først når alle 9 kolonner, rader og kvadrater er ferdig utfylt er spillet ferdig.
Vi kan bruke tallet 0 for å representere en plass som ikke er fylt med et tall enda. For eksempel i

2	0	0	0	0	0	1	3	5

mangler fortsatt sifrene 4, 6, 7, 8, 9 (i kolonnene 2-6).

Oppgave 2a) (4 %)

Lag en enkel funksjon `readOneNumber` som leser inn både rad, kolonne og et tall mellom 1-9. Funksjonen skal skrive ut en pent formatert bekreftelse til brukeren. Du trenger ikke å returnere tallene fra denne funksjonen, og du kan anta at brukeren oppgir kun lovlige verdier. Eksempel på hvordan kjøring av funksjonen skal se ut (grå skrift skal skrives av funksjonen, mens brukeren taster inn de tre tallene, med <Enter> etter hvert tall):

```
>>> readOneNumber()
Rad (1-9): 2
Kolonne (1-9): 3
Tallet (1-9): 4
Posisjon (2,3) inneholder nå 4
```

```
def readOneNumber():
    row = int(input("Rad (1-9): "))
    col = int(input("Kolonne (1-9): "))
    num = int(input("Tallet (1-9): "))
    print("Posisjon ({:d},{:d}) inneholder nå {:d}".format(row, col, num))
```

Oppgave 2b (4 %)

Lag en ny komplett innlesningsfunksjon, kalt `readPositionDigit`.
Denne funksjonen skal ha innparametere «rowNr», «colNr» og «board».
«board» inneholder alle tallene på brettet (se figuren under), og funksjonen skal spørre brukeren om å taste inn en ny verdi for ruten (rowNr, colNr) på tastaturet. Den nye verdien skal lagres i «board», i raden «rowNr» og kolonnen «colNr», og deretter skal «board» returneres fra funksjonen. Du kan anta at brukeren alltid oppgir gyldige verdier i denne del-oppgaven. Eksempel på bruk:

```
>>> readPositionDigit(2,3,[[1,0,0],[2,0,0],[3,0,0]])
Verdi for posisjon (2,3): 8
[[1, 0, 0], [2, 0, 8], [3, 0, 0]]
```

1	0	0…
2	0	8…
3	0	0…
…	…	…

```
def readPositionDigit(rowNr, colNr, board):
    board[rowNr-1][colNr-1] = \
        int(input("Verdi for posisjon ({:d},{:d}): ".format(rowNr, colNr)))
    return board
```

Oppgave 2c (6 %)

Det er viktig at spillebrettet kun skal inneholde tall fra 0-9, og ikke for eksempel «abc». Lag en ny og bedre komplett innlesningsfunksjon «readValidPositionDigit». Funksjonen skal ha samme input og output som over, men også med feilhåndtering denne gangen: Hvis brukeren oppgir noe annet enn et siffer fra 0-9 skal funksjonen skrive ut «Feil! Oppgi et siffer mellom 0 og 9…» og fortsette å spørre om ny verdi helt til brukeren oppgir et siffer mellom 0-9. Eksempel på bruk av funksjonen:

```
>>> readValidPositionDigit(2,3,[[1,0,0],[2,0,0],[3,0,0]])
Verdi for posisjon (2,3): abc
Feil! Oppgi et siffer mellom 0 og 9...
Verdi for posisjon (2,3): [1,2,3]
Feil! Oppgi et siffer mellom 0 og 9...
Verdi for posisjon (2,3): 'a'
Feil! Oppgi et siffer mellom 0 og 9...
Verdi for posisjon (2,3): 10
Feil! Oppgi et siffer mellom 0 og 9...
Verdi for posisjon (2,3): 9
[[1, 0, 0], [2, 0, 9], [3, 0, 0]]
```

1	0	0…
2	0	9…
3	0	0…
…	…	…

Denne oppgave kan løses på flere ulike måter, f.eks.:

```
# med strengmetoden isdigit
def readValidPositionDigit(rowNr, colNr, board):
    num = -1
    while num < 0 or num > 9:
        pos= input("Verdi for posisjon ({:d},{:d}): ".format(rowNr, colNr))
        if not pos.isdigit():
            print("Feil! Oppgi et siffer mellom 0 og 9...")
        else:
            num = int(pos)
            if num not in range (0,10):
                print("Feil! Oppgi et siffer mellom 0 og 9...")
    board[rowNr-1][colNr-1] = num
    return board

# med bruk av Exception
def readValidPositionDigit(rowNr, colNr, board):
    while True:
        try:
            num = int(input("Verdi for posisjon ({:d},{:d}): ".\
                        format(rowNr, colNr)))
        except ValueError:
            pass
        else:
            if num in range (0,10):
                break
        print("Feil! Oppgi et siffer mellom 0 og 9...")
    board[rowNr-1][colNr-1] = num
    return board
```

Oppgave 2d (6 %)

For å løse Sudoku (for eksempel fra et ukeblad), kan vi bruke funksjonen «readValidPositionDigit» både til å lese inn et halvveis utfylt spillebrett og til å fylle inn nye enkelttall. Vi må også sjekke at ikke det samme tallet forekommer to ganger i samme rad, kolonne, eller firkant, men det trenger du ikke ta hensyn til i denne oppgaven.

Lag en funksjon «**readSudokuBoard**» som bruker funksjonen «**readValidPositionDigit**» til å fylle «**board**» med gyldige verdier mellom 0 og 9. Den skal lese inn alle de 9 tallene i kolonne 1 fra toppen til bunnen, før den fortsetter på resten av kolonnene (2-9) fra venstre mot høyre på samme måte, en kolonne om gangen. Output fra funksjonen skal være «**board**» med 81 siffer (0-9).

Eksempel på bruk av funksjonen:

```
>>> readSudokuBoard()
Verdi for posisjon (1,1): 1
Verdi for posisjon (1,2): 2
…
Verdi for posisjon (9,8): 6
Verdi for posisjon (9,9): 7
[[1, 2, 3, 4, 5, 6, 7, 8, 9], [2, 3, 4, 5, 6, 7, 8, 9, 1], [3, 4, 5, 6, 7, 8, 9, 1, 2], ...
 [4, 5, 6, 7, 8, 9, 0, 1, 2], [3, 4, 5, 6, 7, 8, 9, 0, 1], [2, 3, 4, 5, 6, 7, 8, 9, 0], ...
 [1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8], [9, 0, 1, 2, 3, 4, 5, 6, 7]]
```

Husk at man må initiere `board` i denne oppgaven.

```
def readSudokuBoard():
    board = [ [ 0 for i in range(9) ] for j in range(9) ]
    for rowNr in range (1,10):
        for colNr in range(1,10):
            board = readValidPositionDigit(rowNr, colNr, board)
    return board
```

Oppgave 3 – Kodeforståelse (15 %)

Oppgave 3 a) (5 %)

1. Hva blir verdien til `res`, med funksjonen som vist under, etter kommandoen **`>>> res=o3a(6)`**
2. Forklar med en kort setning hva funksjonen gjør.

```
def o3a(n):
    if n == 0:
        a = 0
    elif n == 1:
        a = 1
    else:
        m2 = 0
        m1 = 1
        for i in range(1,n):
            a = m1 + m2
            m2 = m1
            m1 = a
    return a
```

1. `>>>` **`res= o3a(6)`** → **`res = 8`**
2. `o3a` beregner Fibonacci-tall nummer **n** ($x_n = x_{n-1} + x_{n-2}$), der **n** er input til funksjonen.

Oppgave 3 b) (5 %)

1. Hva blir verdien til `res` etter kommandoen **`>>> res = o3b(6)`** med funksjonen under kjøres?
2. Forklar med en kort setning hva funksjonen gjør.

```
def o3b( n ):
    if n == 0:
        a = 1
    else:
        a = n * o3b( n-1 )
    return a
```

1. **`>>> res = o3b(6)`** → **`res = 720`**
2. `o3b` beregner fakultetet av **n** (**n! = n * (n-1) * (n-2) * … * 1**), der **n** er input til funksjonen.

Oppgave 3 c) (5 %)

Vi vil ha en funksjon "**checkRowOK**" som skal teste om ei sudokurad med 9 tall bruker hvert av tallene 1-9 nøyaktig en gang, i tråd med reglene for Sudoku. Eksempel på ønsket utskrift:

```
>>> checkRowOK ( [ 1,2,3,4,5,6,9,8,7 ] )
OK.
>>> checkRowOK ( [ 1,2,3,4,5,2,9,8,7 ] )
Tallet 2 forekommer 2 gonger. Tallet 6 er ikke brukt.
```

"Per" har skrevet følgende funksjon:

```
def checkRowOK (row):
    timesUsed = [0 for x in range(9)]
    for n in row:
        timesUsed[n-1] += 1
    response = ""
    for i in range(9):
        if (timesUsed[i] != 1):
            response += "Tallet " + str(i+1)
            if (timesUsed[i] == 0):
                response += " er ikke brukt. "
            else:
                response += " kommer " + str(timesUsed[i]) + " ganger. "
        else:
            response = "OK."
    print(response)
```

Denne gir riktig output for det første eksemplet over (OK), men gir også OK for det andre eksemplet, på grunn av en feil i funksjonen. Hva er feilen, og hvordan kan den rettes?
(Du trenger ikke å skrive noen kode i denne oppgave, kun angi overfladisk hvor feilen skal fikses.)

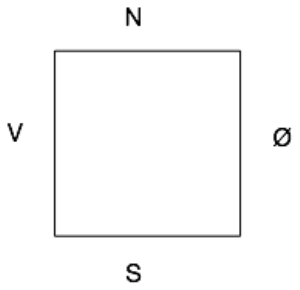
Hvis noe tall er brukt eksakt én gang, så sletter den siste **else**-klausulen alle tidligere verdier (dvs setter strengen **response** til OK). Man må altså tilse at svarstrengen beholder sin verdi også når et tall faktisk forekommer eksakt én gang.

I denne oppgave trenger man ikke å skrive noen kode, men en mulig riktig versjon av funksjonen kan se ut sånn her:

```
def checkRowOK (row):
    timesUsed = [0 for x in range(9)]
    for n in row:
        timesUsed[n-1] += 1
    response = ""
    for i in range(9):
        if (timesUsed[i] != 1):
            response += "Tallet " + str(i+1)
            if (timesUsed[i] == 0):
                response += " er ikke brukt. "
            else:
                response += " kommer " + str(timesUsed[i]) + " ganger. "
    if response == "":
            response = "OK."
    print(response)
```

Oppgave 4: Mer programmering (40 %)

Bridge er et kortspill der man spiller på lag. Lagene kalles Nord/Syd ('N/S') og Øst/Vest ('Ø/V'). Man bruker en vanlig kortstokk med 52 kort, slik at hver spiller har 13 kort. Man melder hvor mange stikk man regner med å ta og hvilken farge som skal være trumf. Meldingene går fra **1 kløver** til **7 grand**. Man skal ta 6 stikk mer (7-13) enn man melder og teksten bak (kløver, ruter, hjerte, spar, evt. grand) angir trumf. Hvis siste melding blir grand skal det spilles uten trumf. Meldingen "3 kløver" betyr altså at man skal ta 9 stikk med kløver som trumf.



Den meldingen man ender opp med (den siste/høyeste) i budrunden kalles en **kontrakt.** Hvis man får flere stikk enn kontrakten tilsier (f.eks. 9 stikk på «2 ruter» som bare krever 8 stikk) kalles de overskytende stikkene for **overstikk.** Hvis man ikke klarer så mange stikk som kontrakten tilsier, går man **beit**.

NB: Det er meningen at man kan/skal bruke funksjoner fra foregående deloppgaver i senere deloppgaver, selv om man kanskje ikke har løst dem helt riktig, og du kan anta at brukeren oppgir kun lovlige verdier.

Oppgave 4a (5 %)

Lag en funksjon med to inn-parametere, «melding» og «antall stikk man har klart», og en returverdi som er «True» hvis man har klart den meldingen man har gitt, ellers «False». Eksempel: bidOk('3 ruter', 10) => True, bidOK('3 ruter', 8) => False.

```
def bidOk(bid, trick):
    return trick >= int(bid.split()[0]) + 6
```

Oppgave 4b (10 %)

Noen kontrakter gir bonuspoeng, f. eks. «**utgang**». En **utgang** er hvis man har meldt og klart 3 grand eller mer (minst 9 stikk, uten trumf), 4 hjerter/spar eller mer (minst 10 stikk med hjerter eller spar som trumf), eller 5 kløver/ruter eller mer (minst 11 stikk med kløver eller ruter som trumf). Lag en funksjon som mottar resultat av et spill (melding og antall stikk) og avgjør om dette er en utgang, og om laget har klart utgangen. Returner True hvis det er en vellykket utgang, ellers returneres False.

```
def game(bid, trick):
    bidlist = bid.split()
    number = int(bidlist[0])
    colour = bidlist[1]
    if colour == 'grand':
        required = 3
    elif colour in ('hjerter', 'spar'):
        required = 4
    elif colour in ('kløver', 'ruter'):
        required = 5
    else:
        print("Ukjent farge")  # dette trenger man ikke å gjøre
        return False
    return number >= required and trick >= number +6
```

Oppgave 4c (15 %)

Poengberegningen i bridge kan være ganske komplisert. Her er en forenklet beskrivelse:

I bridge regnes det ikke poeng for de seks første stikkene. Alle stikk etter det sjette stikket er såkalte trekk, slik at sju stikk tilsvarer ett trekk. Det høyeste du kan melde er da 7 trekk som tilsvarer 13 stikk.

- Med kløver eller ruter som trumf får man 20 poeng pr. tatt trekk (1-7).
- Med hjerter eller spar som trumf får man 30 poeng pr. tatt trekk (1-7).
- I grand (ingen trumf) får man 30 poeng for hvert tatt trekk (1-7) pluss 10 poeng i tillegg.

Man får 50 poeng bonus hvis man har klart en meldt kontrakt. Hvis denne meldingen er en utgang (minst 5 kløver eller ruter, 4 spar eller hjerter, eller 3 grand) får man i stedet 300 poeng i bonus. Hvis man går beit, så får man 0 poeng, men motstanderen får 50 poeng for hvert stikk som manglet for å greie kontrakten.

Lag (to) funksjoner som sammen beregner poengsummen for et spill. Husk at det gis poeng for en trekk og bonuser, eller motstander-poeng for beit.
Hvis laget som spiller kontrakten klarer den, så skal en positiv poengsum returneres. Hvis de går beit, så skal i stedet en negativ poengsum returneres. Eksempler:

```
>>> bridgePoints( '3 ruter', 10)      returnerer 130      (4 * 20 + 50)
>>> bridgePoints( '3 ruter', 8)       returnerer -50      (-50 * 1)
>>> bridgePoints( '3 spar', 12)       returnerer 230      (6 * 30 + 50)
>>> bridgePoints( '4 spar', 12)       returnerer 480      (6 * 30 + 300)
>>> bridgePoints( '4 grand', 12)      returnerer 490      (10 + 6 * 30 + 300)
```

Det er ikke nødvendig å lage akkurat to funksjoner i denne oppgaven, men trolig blir det enklest å lese hvis man gjor det – og lettest å løse blir det sikkert hvis man gjenbruker funksjoner fra oppgavene 4a og 4b.

```
def trickPoints(bid, trick):
    colour = bid.split()[1]
    if colour == 'grand':
        return 10 + 30 * (trick - 6)
    elif colour in ('hjerter', 'spar'):
        return 30 * (trick - 6)
    elif colour in ('kløver', 'ruter'):
        return 20 * (trick - 6)
    else:
        print("Ukjent farge")  # dette trenger man ikke å gjøre

def bridgePoints(bid, trick):
    if game(bid, trick):
        return 300 + trickPoints(bid, trick)
    elif bidOk(bid, trick):
        return 50 + trickPoints(bid, trick)
    else:
        return 50 * (trick - (int(bid.split()[0]) + 6))
```

Oppgave 4d (10 %)

Skriv et program der det registreres flere spill. Bruk funksjonene du har skrevet i oppgave 4 a, b og c (og skriv evt. andre funksjoner du trenger). Hvert spill lagres som en liste som inneholder hvilket lag som fikk kontrakten, meldingen, antall stikk, antall poeng, antall beit-poeng.
Eks: ['N/S', '3 ruter', 9, 110, 0]
Den totale oversikten over spill er dermed en liste bestående av lister som vist:
[['N/S', '3 ruter', 9, 110, 0],
['Ø/V', '3 hjerter', 9, 140, 0],

['N/S', '4 spar', 8, 0, 100],
…]

Når alle spillene er registrert, skal programmet beregne den totale poengsummen for hvert lag (N/S og Ø/V).
Eksempel på utskrift:

```
Lag (N/S eller Ø/V, annet for å slutte): N/S
Melding: 4 grand
Stikk: 9
Lag (N/S eller Ø/V, annet for å slutte): N/S
Melding: 4 grand
Stikk: 10
Lag (N/S eller Ø/V, annet for å slutte): Ø/V
Melding: 5 ruter
Stikk: 11
Lag (N/S eller Ø/V, annet for å slutte):
['N/S', '4 grand', 9, 0, 50]
['N/S', '4 grand', 10, 430, 0]
['Ø/V', '5 ruter', 11, 400, 0]
Total score:
N/S 430
Ø/V 450
```

Denne oppgave går selvsagt å løse på mange ulike måter. Her har vi brukt en hovedfunksjon `bridgeCounter` som leser inn resultatene fra alle spill til en liste og en hjelpfunksjon `calculateTotalPoints` som tar listen og regner ut totalpoeng for de to lagene (N/S og Ø/V).

```
def calculateTotalPoints(resultList):
    pointsNS = 0
    pointsEW = 0
    for round in resultList:
        if round[0] == 'N/S':
            pointsNS += round[3]
            pointsEW += round[4]
        else:
            pointsNS += round[4]
            pointsEW += round[3]
    print("N/S fikk {:d} poeng".format(pointsNS))
    print("Ø/V fikk {:d} poeng".format(pointsEW))

def bridgeCounter():
    resultList = []
    while True:
        jn = input("Registere et til spill? [j/n] ")
        if jn == 'n':
            calculateTotalPoints(resultList)
            return resultList   # trenger ikke å returnere listen
        else:
            team = input("Hvilket lag spiller? ['N/S' / 'Ø/V'] ")
            bid = input("Hvilken melding? ")
            trick = int(input("Hvor mange stikk tog de? [0-13]: "))
            points = bridgePoints(bid, trick)
            if points > 0:
                resultList.append([team, bid, trick, points, 0])
            else:
                resultList.append([team, bid, trick, 0, abs(points)])
```