Løsningsforslag – Eksamenssett 1 (kl. 09:00-13:00)

Oppgave 1 - Teori (25%)

1) Hvilket alternativ er IKKE et lag i TCP/IP-stabelen (stack)?
Riktig svar: Sammenkoblingslaget (Connection layer)

2) Hvilken form for kanalkoding (channel coding) bruker Internettet?
Riktig svar: 16-bit sjekksum (16-bit checksum)

3) Hvilken metode brukes for å håndtere duplikater og at pakker kommer i feil rekkefølge?
Riktig svar: Sequencing (sekvensering)

4) Hva er "replay error" i nettverkssammenheng?
Riktig svar: At en forsinket pakke fra tidligere sesjon blir akseptert i senere sesjon, og at korrekt pakke dermed blir avvist som duplikat.

5) Hvordan håndteres en "replay error"
Riktig svar: Pakkene i en sesjon merkes med en unik ID

6) Hva er det som karakteriserer et DoS (Denial-of-Service) angrep?
Riktig svar: Angriperen sender enorme mengder pakker til en server slik at den ikke kan betjene legitime forespørsler.

7) Hvor mange bits består en IPv6-adresse av?
Riktig svar: 128

8) Hva spesifiserer "1"-ere i en subnett-maske?
Riktig svar: Hvilke bits av IP-adressen som utgjør prefiks.

9) Hva skjer når en melding krypteres?
Riktig svar: Dataene i meldingen endres, slik at kun riktig mottaker kan rekonstruere den opprinnelige meldingen.

10) Hvordan fungerer kryptering med offentlig nøkkel (public key encryption)?
Riktig svar: Hver part får en hemmelig og en offentlig nøkkel. En melding kryptert med en offentlig nøkkel, kan kun dekrypteres med den korresponderende private nøkkelen.

11) Hva var det som var så revolusjonerende med CPU?
Riktig svar: CPU gjorde det mulig å lagre data i maskinens minne som gir store fordeler med tanke på at programmer kan bli mer komplekse og endres fort kun ved å gi nye instruksjoner til minnet.

12) Hva sier Moores lov?
Riktig svar: Loven sier at antall transistorer i en integrert krets dobles hvert 2. år

13) Hvilke fem steg er med i "Fetch/Execute Cycle"?
Riktig svar: Instruction Fetch(IF), Instruction Decode(ID), Data Fetch(DF), Instruction Execute(EX) , Result Return(RR)

14) Hvordan fungerer en transistor?
Riktig svar: En bryter som det enten kan gå strøm gjennom eller ikke, og som man kan styre ved hjelp av strøm.

15) CPU kalles også
Riktig svar: Microprocessor

16) Hvor mange symboler kan representeres med 3 byte?
Riktig svar: 16777216

17) Hva er sant om "run-length-koding"?
Riktig svar: Run-length-koding er tapsløs komprimering, dvs. at den originale representasjonen av 0ere og 1ere kan bli rekonstruert perfekt fra den komprimerte versjonen.

18) Extended ASCII (også kjent som ISO-8859-1) er
Riktig svar: 8 bits kode

19) Hvilket binært tall representeres av det hexadesimale tallet 39A?
Riktig svar: 001110011010

20) Navnet "Bob" skrives som "0100 0010 0110 1111 0110 0010" i Extended ASCII. Hvilket alternativ representerer ordet "obo" i Extended ASCII?
Riktig svar: 0110 1111 0110 0010 0110 1111

Oppgave 2a - Kodeforståelse (5%)

Funksjonen **myst** har følgende kode:

```
def myst(val1, val2):
   if (val1 and val2):
      return 1
   elif (val1 and not val2):
      return 2
   elif (not val1 and val2):
      return 3
   else:
      return 4
```

Hva returneres ved funksjonskallet under?

myst(((True and False) or (False and True)), ((False or True) and (not(not True))))

Riktig svar: 3

Oppgave 2b - Kodeforståelse (5%)

Funksjonen **sqr** skal ta inn en liste (**numbers**) bestående av heltall som parameter. Funksjonen skal endre annethvert heltall i listen (fra og og med andre element) til kvadratet av heltallet (heltallet multiplisert med seg selv). Til slutt skal funksjonen returnere den endrede listen.

Gitt funksjonen:

```
def sqr(numbers):

  #KODE 1
      numbers[x]=numbers[x]**2
   return numbers
```

Eksempel på kall av funksjonen **sqr**:

```
>>> numbers = sqr([2,4,6,8,10,12])
>>> print(numbers)
[2, 16, 6, 64, 10, 144]
```

Hvordan skal linjen med innholdet **#KODE1** i koden til **sqr** se ut for at funksjonen skal fungere på måten beskrevet over ved kjøring?

Riktig svar: for x in range(1,len(numbers),2):

Oppgave 2c - Kodeforståelse (10%)

Funksjonen **prime_numbers** er ment å returnere en liste som inneholder alle primtall, dvs. tall som bare kan deles med seg selv og 1, i et bestemt intervall angitt av parametrene **start** og **stopp**. Anta at *start*alltid vil være større enn 0 og mindre enn *stopp*.

Gitt funksjonen:

```
def prime_numbers(start, stop):
   primes = []
   for num in range(start,stop + 1):
      if num > 1:
         prime = True
         for i in range(2,num):
            #KODE1
               prime = False
               break
         if prime:
```

```
            #KODE2
    return primes
```

Eksempel på kall av funksjonen **prime_numbers:**

```
>>> primes = prime_numbers(1, 16)
>>> print(primes)
[2, 3, 5, 7, 11, 13]
```

Hvordan skal linjene med innholdet **#KODE1**og **#KODE2**i koden til **prime_numbers** se ut for at funksjonen skal ha den tiltenkte virkemåten under kjøring?

Riktig svar:
1) if (num % i) == 0:
2) primes.append(num)

Oppgave 2d - Kodeforståelse (5%)

Funksjonen **palindrome** tar inn en streng som parameter og har til hensikt å sjekke om ordet eller uttrykket representert i strengen gir samme resultat enten det leses fra høyre eller venstre. Hvis ordet eller utrykket gir samme resultat skal funksjonen returnere *True*. Hvis ikke skal den returnere *False*.

Funksjonen skal kun returnere *True*i tilfeller hvor tegnsetting i strengen blir helt lik uavhengig av hvilken vei strengen leses. Det betyr at funksjonskallet **palindrome("Radar")**vil returnere *False* siden funksjonen skiller på store og små bokstaver.

Gitt funksjonen:

```
def palindrome(s):
    #KODE1
```

Eksempel på kjøring av funksjonen **palindrome**:

```
>>> palindrome("radar")
True
```

Hvordan skal linjen med innholdet **#KODE1** (return-setningen) i koden til **palindrome** se ut for at funksjonen skal fungere som beskrevet over ved kjøring?

Riktig svar: return not(bool(s.find(s[::-1])))

Oppgave 2e - Kodeforståelse (5%)

Gitt funksjonen:

```
def myst(tall1,tall2,nr):
    for i in range(2,nr):
        nytt = tall1 + tall2
        tall1 = tall2
        tall2 = nytt
    return nytt
```

Hva skrives ut når følgende kode kjøres?

print(myst(0,1,7))

Riktig svar: 8

Oppgave 2f - Kodeforståelse (5%)

Gitt funksjonen:

```
def myst(x,y):
    if x%y == 0:
        return y
    else:
        return myst(y,x%y)
```

Hva skrives ut når følgende kode kjøres?

print(myst(24,88))

Riktig svar: 8

Oppgave 3a – Programmering (5%)

Skriv funksjonen **read_file** som tar en inn-parameter **filename**. Denne funksjonen skal lese inn en tekstfil **filename**, som bekriver resultatene fra årets flerkamp-konkurranse. Resultatene er lagret i tekstfilen *flerkamp.txt*, som ligger i den samme mappen (directory) som du skal lagre python-koden. Resultatene i tekstfilen er på formatet beskrevet over. Funksjonen skal returnere innholdet i tekstfilen i form av én lang streng. Dersom filen ikke finnes skal funksjonen skrive ut feilmeldingen "Kan ikke finne filen flerkamp.txt" til skjermen, og returnere verdien *None*.

Eksempel på kjøring av funksjonen og utskrift av returverdi:

```
>>> resultater = read_file('flerkamp.txt')

>>> resultater
```

'Name, Poker, Highjump, Balloonshooting,
SausageEating, HoldBreath\nJohn, 8, 1.67, 17, 23, 2:01.65\nLisa, 12, 1.3
0, 12, 13, 1:13.02\nPer, 8, 1.55, 8, 0, 1:51.35\nNelly, 2,
1.34, 9, 17, 0:31.18\nNora, 5, 1.87, 13, 5, 2:01.65\n'

```
# Oppgave 3A: 5%
def read_file(file):

    # First we need to check if the filename exists. If not, ask for a new name.
    try:
        open_file=open(file, 'r') # Read only mode
        textResults = open_file.read()
        open_file.close() # Closing to save memory
    except IOError: # No such file.
        print("'"+file+"' could not be found.")
        return

#    print('fil:',textResults)
    return textResults
```

Oppgave 3b – Programmering (5%)

Skriv funksjonen **list_from_string** som tar inn strengen **txt** som inn-parameter. Forutsett at **txt** har et format tilsvarende én enkelt linje i *flerkamp.txt*, dvs. et sett med verdier (navn og resultat per øvelse) separert med komma og et vilkårlig antall mellomrom (whitespace). Funksjonen skal returnere en liste med strenger som beskriver de enkelte verdiene. Eventuelle mellomrom, linjeskift eller tabulatorer (whitespace) i strengen må fjernes fra hvert element i listen før listen returneres.

Eksempel på bruk:

```
>>> list_from_string("Lisa, 12, 1.30, 12, 13, 1:13.02\n")
['Lisa', '12', '1.30', '12', '13', '1:13.02']
```

```
# Oppgave 3B: 5%
def list_from_string(streng):
    liste = streng.split(",")
    for i in range(len(liste)):
        liste[i] = liste[i].strip()
#    print('liste:',liste)
    return liste
```

Oppgave 3c – Programmering (5%)

Skriv funksjonen **make_result_list** som tar inn strengen returnert av funksjonen **read_file** (Oppgave 3a) som inn-parameter. Funksjonen **make_result_list** skal returnere en to-dimensjonal liste der hvert listeelement er en liste som innholder verdiene fra hver linje i filen*flerkamp.txt.* Bruk gjerne funksjonen **list_from_string** (Oppgave 3b) i løsningen din.

Eksempel på bruk:

>>> string = read_file('flerkamp.txt')

>>> string
'Name, Poker, Highjump, Balloonshooting,
SausageEating, HoldBreath\nJohn, 8, 1.67, 17, 23, 2:01.65\nLisa, 12, 1.3
0, 12, 13, 1:13.02\nPer, 8, 1.55, 8, 0, 1:51.35\nNelly, 2,
1.34, 9, 17, 0:31.18\nNora, 5, 1.87, 13, 5, 2:01.65'

>>> results = make_result_list(string)

>>> results

[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', '8', '1.67', '17', '23', '2:01.65'], ['Lisa', '12', '1.30', '12', '13', '1:13.02'], ['Per', '8', '1.55', '8', '0', '1:51.35'], ['Nelly', '2', '1.34', '9', '17', '0:31.18'], ['Nora', '5', '1.87', '13', '5', '2:01.65']]

```
# Oppgave 3C: 5%
def make_result_list(string):
    print()
    my_list = []
    in_list = string.split("\n")
    for line in in_list:
        my_list.append(list_from_string(line))
    return my_list
```

Oppgave 3d – Programmering (6%)

Skriv funksjonen **time_to_seconds** som tar strengen **time** som inn-parameter. Strengen angir en deltakers sluttid i en øvelse (f.eks. holde pusten lengst mulig) og vil ha følgende format: *min:sek.hundredeler*. Funksjonen skal gjøre om strengen til et flyttall med formatet *sekunder.hundredeler* og returnere dette flyttallet.

Eksempel på bruk:

>>> time_to_seconds('2:01.65')

121.65

>>> print(type(time_to_seconds('2:21.65')))

<class 'float'>

```
# Oppgave 3D: 6%
# In: a string with current format: minutes:seconds.hundrethsofseconds
# out: seconds.hundreds (as float)
def time_to_seconds(time):
    minute = int(time.split(':')[0])
    sec = int(time.split(':')[1].split('.')[0])
    hs = time.split(':')[1].split('.')[1]
    return float(str(str(minute*60+sec)+'.'+hs))
```

Oppgave 3e – Programmering (9%)

Skriv funksjonen **str_to_numbers** som tar inn **results**, en to-dimensjonal liste, som parameter. Funksjonen skal kunne ta inn den to-dimensjonale listen som returneres fra funksjonen **make_result_list** (Oppgave 3c) og gjøre om tall på strengformat til heltall eller flyttall. Hvis et element har formatet '2:23.56' skal disse gjøres om til antall sekunder og antall hundredeler (altså her 143.56 som flyttall). Hvis elementet har formatet '1.3' skal det gjøres om til et flyttall, mens hvis strengen bare inneholder et tall som '12' skal den gjøres om til et heltall. Funksjonen **str_to_numbers** skal returnere den formaterte (to-dimensjonale) listen.

Eksempel på bruk:

>>> # results inneholder resultatet fra oppgave 3c, og den har følgende format:

>>> results # Dette er formatet fra oppgave 3c
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', '8', '1.67', '17', '23', '2:01.65'], ['Lisa', '12', '1.30', '12', '13', '1:13.02'], ['Per', '8', '1.55', '8', '0', '1:51.35'], ['Nelly', '2', '1.34', '9', '17', '0:31.18'], ['Nora', '5', '1.87', '13', '5', '2:01.65']]

>>> results = str_to_number(results)

>>> results # Dette er results fra oppgave 3e

[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'],

['John', 8, 1.67, 17, 23, 121.65], ['Lisa', 12, 1.3, 12, 13, 73.02],

['Per', 8, 1.55, 8, 0, 111.35], ['Nelly', 2, 1.34, 9, 17, 31.18],

['Nora', 5, 1.87, 13, 5, 121.65]]

```
# Oppgave 3E: 9%
def str_to_numbers(liste):
    for row in range(len(liste)):

        # If the element has both ':' and '.', we need to secondify it.
        for item in range(len(liste[row])):
            if (liste[row][item].find(":") != -1):
#                print(liste[row][item])
                liste[row][item] = time_to_seconds(liste[row][item])
            # If it is only has a '.', it is a float.
            elif (liste[row][item].find(".") != -1) :
                liste[row][item] = float(liste[row][item])
            # If all chars in the element is a digit, we intify it.
            elif liste[row][item].isdigit():
                liste[row][item] = int(liste[row][item])
    return liste
```

Oppgave 3f – Programmering (9%)

Skriv funksjonen **find_data**, som har input-parametrene **event**, **name** og **results**. Parametrene inneholder, i den rekkefølgen, en øvelse, deltagerens navn og resultatlisten for en fullstendig flerkamps-konkurranse. Denne listen er formattert slik som den returneres fra funksjonen**str_to_numbers** (i oppgave 3e). Navnet på øvelsene og deltagerne er i ukjent rekkefølge.

Funksjonen skal bruke listen til å finne resultatet for en deltager i den oppgitte øvelsen, og returnere denne verdien.

I eksempelet under er det vist hvordan resultatet (returverdien fra **str_to_numbers**) ser ut før **find_data** blir kjørt:

>>> results
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', 8, 1.67, 17, 23, 121.65], ['Lisa', 12, 1.3, 12, 13, 73.02], ['Per', 8, 1.55, 8, 0, 111.35], ['Nelly', 2, 1.34, 9, 17, 31.18], ['Nora', 5, 1.87, 13, 5, 121.65]]

>>> find_data('SausageEating','John',results)

23

```
# Oppgave 3F: 5%
# The first row (0) contains names of events. If we get the correct column
# we can then traverse each line in search of the correct name.
def find_data(navn, event, results):
    column = results[0].index(navn)
    for i in results:
        if i[0] == event:
            return i[column]
```

Oppgave 3g – Programmering (5%)

Skriv funksjonen **event_results** som tar inn-parametrene **event** og **results**. Parameteren **event** er en streng som angir én bestemt øvelse (f.eks. *Poker*), mens results vil være datastrukturen som returneres av funksjonen **str_to_numbers** (Oppgave 3e). Funksjonen **event_results** skal returnere en to-dimensonal liste, slik som i eksemplet nedenfor. Listen skal være sortert etter resultat slik at vinneren av den angitte øvelsen og hans/hennes resultat skal kommer først i listen, mens taperen og hans/hennes resultat kommer sist.

Du må gjerne bruke den eksisterende funksjonen **sort_list** hvis du ønsker det. Den står beskrevet i starten av oppgave 3, men oppsummert sorterer den en todimensjonal liste list på elementnummer elem:

>>> sort_list([['John', 8], ['Lisa', 12], ['Per', 8]],1)

[['Lisa', 12], ['John', 8], ['Per', 8]]

>>> results # Fra oppgave 3e
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', 8, 1.67, 17, 23, 121.65], ['Lisa', 12, 1.3, 12, 13, 73.02], ['Per', 8, 1.55, 8, 0, 111.35], ['Nelly', 2, 1.34, 9, 17, 31.18], ['Nora', 5, 1.87, 13, 5, 121.65]]

>>> event_results('Poker',results)

[['Lisa', 12], ['John', 8], ['Per', 8], ['Nora', 5], ['Nelly', 2]

```python
# Oppgave 3G: 5%
# Sorts one event based on results, lower value higher
# return a list of lists, with [name, value].
def event_results(event, results):
    column = results[0].index(event)
    event_result = []
    for person in results[1:]: # Skip the first, as that's the header
        event_result.append([person[0],person[column]])
    # Now for the sorting, using the specified function sort_list
    return sort_list(event_result,1)
```

Hjelpefunksjon:

```python
# Helper function: sorts a list of lists based on its second value
# Will be specified at the exam. Values sorted lowest first, increasing
def sort_list(liste,column):
    return sorted(liste,key=lambda l:l[column], reverse=True)
```

Flerkamp.txt:

```
Name,   Poker,  Highjump,  Balloonshooting, SausageEating,  HoldBreath
John,   8,      1.67,     17,          23,          2:01.65
Lisa,   12,     1.30,     12,          13,          1:13.02
Per,    8,      1.55,     8,           0,           1:51.35
Nelly,  2,      1.34,     9,           17,          0:31.18
Nora,   5,      1.87,     13,          5,           2:01.65
```