Løsningsforslag – Eksamenssett 2 (kl. 15:00-19:00)

Oppgave 1 - Teori (25%)

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

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

3) 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.

4) Hva er "jitter" innenfor nettverk?
Riktig svar: Endringer i forsinkelser og lengden på forsinkelsene.

5) Hva går "buffer overflow" ut på?
Riktig svar: At det oversendes mer data enn hva mottaker forventer. Overskrider en databuffers grenser og skriver til nabolokasjoner i minnet. Kan føre til minneaksessproblemer, feil resultater og krasj.

6) Hva er "wiretapping"?
Riktig svar: At en angriper kopiere datapakker som traverserer nettet for å få tak i informasjon.

7) Hva er sant om linje- og pakke-svitsjing?
Riktig svar: I linje-svitsjing opprettes koblingen mellom partene ved behov, og avsluttes etter endt bruk.

8) Hvor mange bit består en IPv4-adresse av?
Riktig svar: 32

9) Hva identifiserer prefikset i en IPv4-adresse?
Riktig svar: Det unike fysiske nettverket en datamaskin er tilknyttet.

10) Hva er en sub-nett maske (addresse maske)?
Riktig svar: En verdi som spesifiserer den eksakte grensen mellom prefikset og suffikset i subnetting og klasseløs adressering.

11) Hva kalles mindre kretskort som plugges inn i hoved(krets-)kortet?
Riktig svar: Datterkort

12) Hva er Photolithography (fotolitografi)?
Riktig svar: En teknikk som har gjort at integrerte kretser har blitt enklere å lage. Ledninger printes på chipene i flere lag.

13) Hva er riktig om harddisker?
Riktig svar: Permanent og random access.

14) Hva er hovedoppgaven til ALU?
Riktig svar: Utføre regneoperasjoner

15) Hvilke 2 registre finner vi i kontrollenheten?
Riktig svar: Programteller og instruksjonsregister

16) Hvilket tall i titallssystemet representerer det binære tallet 100111?
Riktig svar: 39

17) Hvilken webfargekode (heksadesimalfargekode) representerer hvit?
Riktig svar: #FFFFFF

18) Informasjon som beskriver informasjon kalles:
Riktig svar: metadata

19) Hva sier Nyquist-regelen for sampling?
Riktig svar: Nyquist-regelen sier at samplingsfrekvensen må være minst dobbelt så rask som den raskeste frekvensen. Ettersom menneskelige ører kan høre lyder opp til ca. 20 000Hz, vil samplingsfrekvens på 40 000Hz oppfylle Nyquists regel for digital lydopptak.

20) Hva er sampling?
Riktig svar: Å ta målinger på regelmessige intervall

Oppgave 2a - Kodeforståelse (5%)

Gitt følgende funksjon:

```
def myst(val1, val2):
    if (val1 and val2):
        return 'a'
    elif (val1 and not val2):
        return 'b'
    elif (not val1 and val2):
        return 'c'
    else:
        return 'd'
```

Hva returneres ved funksjonskallet under?

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

Riktig svar: d

Oppgave 2b - Kodeforståelse (5%)

Gitt funksjonen under:

```
def myst(numbers, flag):
   n = []
   for i in numbers:
      if (flag and (i%2 == 0)) or (not flag and (i%2 != 0)):
         if i not in n:
            n.append(i)
      n.sort()
   return n
```

Hva skrives ut når følgende funksjonskall kjøres:
print(myst([1,8,6,2,5,4,9,4,8,0], True))

Riktig svar: [0, 2, 4, 6, 8]

Oppgave 2c - Kodeforståelse (5%)

Funksjonen **count_items** tar inn ei liste med navn på personer (tekststrenger) og teller antall forekomster av hvert navn i lista. Funksjonen skal returnere en *dictionary* hvor hvert navn i lista er representert én gang og som inneholder informasjon hvor mange ganger navnet forekommer i lista.

```
def count_items(names)
 d = {}
 for name in names
   KODE#1
 return d
```

Eksempel på kall av funksjonen count_items:

```
>>> print(count_items(['Anna', 'John', 'John', 'Thomas', 'Jane', 'Anna', 'Lis']))
{'Anna': 2, 'John': 2, 'Thomas': 1, 'Jane': 1, 'Lis': 1}
```

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

Riktig svar: d[name] = d.get(name,0)+1

Oppgave 2d - Kodeforståelse (5%)

Gitt funksjonen:

```
def myst(t1,t2):
   n = 2
```

```
    while (t1 % n != 0) or (t2 % n != 0):
        n+= 1
    return n
```

Hva skrives ut når følgende kode kjøres?

print(myst(49,42))

Riktig svar: 7

Oppgave 2e - Kodeforståelse (10%)

Funksjonen **bin_search** beskrevet under er ment å utføre et iterativt binærsøk:

```
def bin_search(values, val, imin, imax):
    while imin < imax:
      #KODE1
       if #KODE2
          return True
       elif #KODE3
          imin = imid+1
       else:
          imax = imid-1
    return False
```

Eksempel på kall av funksjonen **bin_search**:

```
>>> A=[1,2,3,9,11,13,17,25,57,90]
>>> print(bin_search(A,57,0,len(A)-1))
True
```

Hva er riktig kode for **#KODE1**, **#KODE2**og **#KODE3**?

Riktig svar: 1) imid = (imin+imax)//2 2) val == liste[imid]: 3) val > values[imid]:

Oppgave 2f - Kodeforståelse (5%)

Funksjonen **sumofn** (beskrevet under) er ment å være en rekursiv funksjon som beregner summen av alle heltall fra og med 1 til og med verdien funksjonen tar inn.

```
def sumofn(n):
    if n==1:
        return 1
    #Kode1
```

Eksempel på kall av funksjonen **sumofn**:

```
>>> sumofn(5)
15
```

Hvordan skal linjen merket **#Kode1** i funksjonen **sumofn** se ut for at funksjonen skal fungerer slik beskrevet over?

Riktig svar: return n+sumofn(n-1)

Oppgave 3a – Programmering (5%)

Skriv funksjonen **menu** som skal skrive ut til skjerm ulike oppgaver auksjonsprogrammet støtter, og la brukeren taste inn (via tastatur) hvilken oppgave han/hun vil ha gjennomført. Funksjonen har ingen parametre. Eksemplet under viser hvordan menyen skal se ut når funksjonen **menu** kalles. Dersom brukeren taster et gyldig valg (dvs. tegnet *i*, *s*eller *q* – se eksemplet under) skal funksjonen returnere en streng bestående av tegnet som ble tastet. I motsatt tilfelle skal funksjonen returnere *None*.

Eksempel på bruk av funksjonen:

>>> menu()

Select task:

'i' to show the highest bid of an item,

's' to save all winning bids to file,

'q' to exit.

Choice:

```
# The solutions provided here are examples. We make a point in keeping the code
# simple
and understandable rather than as compact and cool as possible.


# auction.py has to sit in the same directory as this exam code. The code for
# auction.py can be found at the bottom of this file

import auction # code at the bottom of this file

# Oppgave 3A: 5%
def menu():
    choice = input("'i' to show the top bid on an item,\n"+
                       "'s' to save all winning bids to file,\n"+
                       "'q' to exit.\n"+
                       "Choice: ")

    #   return choice if choice in 'isq' else None # Denne er kortest.

    if choice in ['i','s','q']: # or in 'isq'
        return choice
    else:
        return None # Or return, or pass. Or just omit return as the result
                    # will be None nonetheless
```

Oppgave 3b

Skriv funksjonen **item_offers** som har inn-parametrene **item** og **data**. Parameteren **item** er en en streng angir navnet på en auksjonsgjenstand (f.eks. *tv*). Parameteren **data** angir en to-dimensjonal liste med auksjonsdata formatert som beskrevet i den innledende oppgavebeskrivelsen (også vist først i eksemplet nedenfor). Funksjonen skal returnere en to-dimensjonal liste som inneholder navnet til budgiverne og budene de har gitt på den angitte auksjonsgjenstanden (se under for eksempel på struktur).

Eksempel på bruk:

>>> data
[['Customer', 'vase', 'maleri', 'sykkel', 'lego', 'tv'], ['Per', 100, 0, 200, 0, 1500], ['Ida', 110, 50, 200, 0, 1500], ['Ottar', 200, 600, 200, 0, 1700], ['Dag', 200, 600, 200, 0, 1700], ['Lise', 400, 600, 0, 0, 0]]

>>> item_offers('tv',data)

[['Per', 1500], ['Ida', 1500], ['Ottar', 1700], ['Dag', 1700], ['Lise', 0]]

```
# Oppgave 3B: 10%
# A two-dimensional list over who bids what on each item.
def item_offers(item,data):
##    print(data)
    # Which column is the item in? Need to look it up.
    column = data[0].index(item)

    # where to store the two-dimensional list to return:
    offers = []

    # Need to remove the first row, as it contains the names.
    # Then stepwise add a new list with bidders name and bid amount.
    for bid in data[1:]:
        offers.append([bid[0],bid[column]])
    return offers
```

Oppgave 3c

Skriv funksjonen **item_winner** som tar inn-parametrene **item**og **data**. Parameteren item er en streng som angir navnet på en auksjonsgjenstand (f.eks. *tv*). Parameteren **data** angir en to-dimensjonal liste med auksjonsdata formatert som beskrevet i den innledende oppgavebeskrivelsen. Funksjonen skal returnere en liste som inneholder to elementer: Navnet på budgiveren med det høyeste budet og hvor høyt budet var. Dersom det ikke eksisterer et bud på en gitt gjenstand skal funksjonen returnere en tom liste. Forutsett at parameteren **item** alltid identifiserer en gjenstand som finnes på auksjonen.

Bruk gjerne funksjonen **sort_list** beskrevet i den innledende oppgavebeskrivelsen (bruken er illustrert nedfor) i forbindelse med sortering.

Dersom flere budgivere har budt høyeste verdi er vinneren den første – det er denne budgiveren som ga budet først hvis du bruker sort_list.

Eksempel på bruk av funksjonen **sort_list.**

>>> auction.sort_list([['a',3],['b',12],['c',10]] , 1) # Du trenger ikke programmere denne.

[['b',12], ['c',10], ['a',3]]

Eksempel på bruk av funksjonen **item_winner**:

>>> data
[['Customer', 'vase', 'maleri', 'sykkel', 'lego', 'tv'], ['Per', 100, 0, 200, 0, 1500], ['Ida', 110, 50, 200, 0, 1500], ['Ottar', 200, 600, 200, 0, 1700], ['Dag', 200, 600, 200, 0, 1700], ['Lise', 400, 600, 0, 0, 0]]

>>> item_winner('tv', data)

['Ottar', 1700]

```python
# Oppgave 3C: 10%
# Find the highest bidder on an item
# The first name with the top bid gave the first bid, so that's the winner
def item_winner(item, data):

    # First find the list of all bidders on an item
    offers = item_offers(item, data)
    # Sort them according to the second value
    # Exam descriptions states that this will sort so the first bid with
    # top value comes first. So we just use it.
    # As stated on exam day: a typo in the function list specified that sort_list
    # sorted by lowest value first. That was not the case, instead follow the
    # example.
    offers = auction.sort_list(offers,1)

    # If there are no bid on an item, return empty list:
    if offers[0][1] == 0:
        return []

    # Define the winners name
    winner = offers[0][0]
    bid = offers[0][1]
    return [winner,bid]
```

Oppgave 3d

Skriv funksjonen **all_winners_dict** som tar inn-parameteren **data**, som er en todimensjonal liste formatert som beskrevet i innledningen til programeringsoppgavene (også illustrert under). Funksjonen skal finne ut hvilken budgiver som har gitt det høyeste budet for hver enkelt auksjonsgjenstand. Funksjonen skal returnere en dictionary. Denne skal bruke navnet på hver gjenstand som nøkkel. Til hver nøkkel skal det knyttes en verdi i form av en liste bestående av to elementer: (1) navnet på budgiveren (vinneren av auksjonen) og (2) vinnerbudet.
Eksempel på bruk:

>>> data
[['Customer', 'vase', 'maleri', 'sykkel', 'lego', 'tv'], ['Per', 100, 0, 200, 0, 1500], ['Ida', 110, 50, 200, 0, 1500], ['Ottar', 200, 600, 200, 0, 1700], ['Dag', 200, 600, 200, 0, 1700], ['Lise', 400, 600, 0, 0, 0]]

>>> all_winners_dict(data)

{'vase': ['Lise', 400], 'painting': ['Ottar', 600], 'bicycle': ['Per', 200], 'lego': [], 'tv': ['Ottar', 1700]}

```python
# Oppgave 3D: 5%
# Find winners for all items on auction, and add them to a dictionary
def all_winners_dict(data): # save as dictionary

    winners_dict = {}
    # First iterate through all items, and we need to find out how many items
    # there are
    # We need to remember that we can't run on the first item (0) in the list,
    # as that is the name
    for i in range(1,len(data[0])):
        item = data[0][i]
        winners_dict[item] = item_winner(item, data)

    return winners_dict
```

Oppgave 3e

Skriv funksjonen **save_auction_data** som tar inn-parameteren **data**. Parameteren data er en to-dimensjonal liste med auksjonsdata formatert slik som beskrevet i den innledende beskrivelsen til Oppgave 3. Funksjonen skal lagre resultatet (return-verdien) av funksjonskallet **all_winners_dict(data)** til binærfilen *auction_winners.db* som ligger i den samme mappen/katalogen som auksjonsprogrammet. Brukeren skal få beskjed om hvorvidt det har lyktes å skrive dataene til binærfilen eller ikke (se eksempel under).

Eksempel på vellykket lagring:

>>> save_auction_data(data)

Dictionary data successfully written to file.

Eksempel på feil ved lagring:

>>> save_auction_data(data)

Could not write dictionary data to file.

Skriv funksjonen **save_auction_data** som tar inn-parameteren **data**. Parameteren data er en to-dimensjonal liste med auksjonsdata formatert slik som beskrevet i den innledende beskrivelsen til Oppgave 3. Funksjonen skal lagre resultatet (return-verdien) av funksjonskallet **all_winners_dict(data)** til binærfilen *auction_winners.db* som ligger i den samme mappen/katalogen som auksjonsprogrammet. Brukeren skal få beskjed om hvorvidt det har lyktes å skrive dataene til binærfilen eller ikke (se eksempel under).

Eksempel på vellykket lagring:

>>> save_auction_data(data)

Dictionary data successfully written to file.

Eksempel på feil ved lagring:

>>> save_auction_data(data)

Could not write dictionary data to file.

```python
# Oppgave 3E: 5%
# Save a dictionary to a binary file:
def save_auction_data(data): # 5%
    dict = all_winners_dict(data)
    import pickle
    try:
        f = open("auction_winners.db",'wb')
        pickle.dump(dict,f)
        f.close()
    except:
        print("Could not write dictionary data to file.")
    else:
        print("Dictionary data successfully written to file.")
```

Oppgave 3f

Skriv funksjonen **task**. Funksjonen skal først opprette en variabel *data*, og sette denne til resultatet av kallet til funksjonen **get_auction_data('auctiondata.txt')**fra modulen*auction.*

Deretter skal funksjonen kalle **menu**og lagre returverdien fra denne i en variabel. Hva funksjonen **task** så skal gjøre avhenger av verdien til variabelen:

> *'i'*: Brukeren skal bli bedt om å skrive inn navnet på en auksjonsgjenstand. Deretter skal funksjonen kalle **item_winner** (Oppgave 3c) med navnet på auksjonsgjenstanden, og variabelen *data* som argument. Funksjonen skal så skrive ut hvem som har vunnet gjenstanden, og hva budet var (eksempelvis *Vinner av tv er Ottar som bød 1700.*)
>
> Hvis det ikke har kommet inn noe bud på en gjenstand (som bilbanen i eksempelet) skal det skrives ut *Ingen bud på lego.*
>
> '*s*': Funksjonen skal kalle **save_auction_data** (Oppgave 3e) med variabelen *data*som argument.
>
> '*q*': Programmet skal avsluttes.

Menyen skal presenteres igjen helt til brukeren velger å avslutte med *q*.

```python
# Oppgave 3F: 5%
# Making a menu for accessing all kinds of functions that are made
def task(): # 5%
    data = auction.get_auction_data('')
    fortsette = True
    while fortsette:

        choice = menu()
        if choice == 'i':
            item = input("Name item: ")
            vinner = item_winner(item,data)
            if vinner: # The list is not empty
                print('Vinner av',item,'er',vinner[0],'som bød',vinner[1])
            else:
                print("No such item on auction:",item)
        elif choice == 's':
            save_auction_data(data)
        if choice == 'q':
            fortsette = False
```

```python
'''
The following code is code that should be added to a file 'auction.py' and sit in
the
same folder as your exam solution. It is to be imported as a module.
'''

''' # start auction.py
# auction.py
# The module auction.py contains some helper functions.

# get_auction_data _really_ uses filename to load all auction
# values from file into a two-dimensional list. We won't show
# the complete code here, but instead return a set of values
# that represents the kind of data one would get. The data
# fits the examples given in the assignments.
def get_auction_data(filename):
    return [['Customer','vase','maleri','sykkel','bilbane','tv'],
            ['Per',100,0,200,0,1500],
            ['Ida',110,50,200,0,1500],
            ['Ottar',200,600,200,0,1700],
            ['Dag',200,600,200,0,1700],
            ['Lise',400,600,0,0,0]]

# Bid returns the bid from customer on item, given dataset data.
def bid(customer, item, data):
    column = data[0].index(item)
    for i in data:
        if i[0] == customer:
            return i[column]

# Helper function: sorts a list of lists based on its second value
# Values sorted lowest first, increasing
def sort_list(liste,column):
    return sorted(liste,key=lambda l:l[column], reverse=True)

''' # End auction.py
```