Løsningsforslag, TDT4110, 10/12 kl 1500

Oppgave 1 - Teori

Hva står CPU for?

Central Processing Unit.

Hva er cache?

En veldig rask minneteknologi

Hvilke 5 steg er med i "Fetch/Execute Cycle"?

Instruction Fetch(IF), Data Fetch(DF), Instruction Decode(ID), Result Return(RR), Instruction Execute(EX)

Hva er pipelining?

En teknikk der en CPU kan utføre flere instruksjoner parallellt

Hvilke av disse lagringsenhetene er IKKE en sekundærlagringsenhet?

Hurtigbufferet i datamaskinen.

Hvilket heksadesimalt tall representerer $(100111)_2$.

27.

Navnet “Bob” skrives som “0100 0010 0110 1111 0110 0010” i Extended ASCII. Hvilket alternativ representerer ordet “obo” i Extended ASCII?

0110 1111 0110 0010 0110 1111

Informasjon som beskriver informasjon er kallt

Metadata

Et bilde har 800*600 piksler, i 16 bit fargeformat. Hvor mye plass trenger det ukomprimert?

Omtrent 1 MB

Hvilken av følgende komprimeringer er loss-less?

Run-length coding

Hva er phishing

Å opptre som en kjent nettside (f.eks. nettbank) for å få tak i personlig informasjon som f.eks. aksesskoder, kontonummer, etc.

Hva er scams

Å lure brukere av f.eks. en nettside eller en tjeneste til å investere penger eller gjøre noe ulovlig.

Hvordan fungerer replay-angrep?

Pakker fra tidligere sesjoner fanges opp og sendes. For eksempel passord-pakker fra tidligere påloggінger.

Hva skjer når en melding krypteres?

Dataene i meldingen endres, slik at kun riktig mottaker kan rekonstruere den opprinnelige meldingen.

Hva er riktig om Payload encryption?

Krypterer kun meldingsinnholdet i pakken og ikke selve pakkehodet.

Hva er sant om transportlaget?

Transportlaget sørger for at all data blir levert slik den ble sendt; komplett og i riktig rekkefølge.

Hva står betegnelsen WAN for i forbindelse med nettverk?

Wide Area Network

Hva vil universal service si i forbindelse med nettverk?

Å tillate kommunikasjon mellom datamaskiner uavhengig av hvilken type nettverk de sitter på.

Hva er masken til IPv4-adressen 255.255.128.0 i CIDR-notasjon

/17

Hva står TCP for i nettverkssammenheng?

Transmission Control Protocol

Oppgave 2 - Kodeforståelse

2a

Hva skrives ut av følgende kode?

('Anne', {'Per': '82.50', 'Anne': '73.00', 'Svein': '93.50'})

```
def myst3(liste,navn):
    dikt = {}
    for i in liste:
        dikt[i[0]] = f'{(sum(i[1:])/(len(i)-1)):.2f}'
    return navn,dikt

liste = [['Per',98,67],['Anne',80,66],['Svein',99,88]]
print(f'{myst3(liste,"Anne")}')
```

2b

Hva skrives ut av denne kodebiten?

[17,19,65,82,99]

```
def myst(a):
    b = []
    while len(a) > 0:
        c = a[0]
        for i in a:
            if i < c:
                c = i
        a.remove(c)
        b.append(c)
    return b


print(myst([99,17,65,19,82]))
```

2c

Hva skrives ut av denne koden?

2

```
def myst(a,b):
    while b != 0:
        c = b
        b = a % b
        a = c
    return a

print(myst(30,4))
```

2d

Hva skrives ut av denne koden?

00010010

```
def myst2(a,b):
    while len(a)%b != 0:
        a = '0' + a
    return a


def myst(a,b,c):
    d = ''
    while a > 0:
        d = str(a%b) + d
        a = a//b
    return myst2(d,c)

print(myst(18,2,4))
```

2e

Hva skrives ut av denne koden?

1

```
def myst(mat):
    r = len(mat)
    c = len(mat[0])
    for i in range(r):
        for j in range(c):
            if mat[i][j] >= 6:
                mat[i][j] = 0
            else:
                mat[i][j] = 1
    return sum(mat[1])

print(myst([[3, 2, 0], [19, 0, 18], [0, 6, 0]]))
```

2f

Hva skrives ut av denne koden?

68

```
def myst1(x,y,z):
    t = x
    x = y
    y = t
    for i in range(z):
        z = x + y
        y*= 2
    return z

print(myst1(2,4,6))
```

2g

Hva skrives ut av denne koden?

10010

```
def myst(a,b,c):
    d = ''
    while a > 0:
        d = str(a%b) + d
        a = a//b
    return d

print(myst(18,2,4))
```

2h

Hva må a og b være for at programmet skal skrive ut MARTIN?

a = 3, b = 6

```
def myst(tekst,a,b):
    return tekst[a::b].upper()

print(myst('mgfmiyaieabarbnramisdtnaooeiehnnrnza',a,b))
```

2i

Hva skrives ut av denne koden?

['e',7]

```
def myst(a,b):
    b = [[a[0],1]]
    for i in range(1,len(a)):
        funnet = False
        for j in b:
            if a[i] == j[0]:
                j[1] += 1
                funnet = True
        if not funnet:
            b.append([a[i],1])
    return b

tekst = 'DettE er en enkel streng. Eller to'
b = []
b = myst(tekst,b)
print(b[1])
```

2j

Hva skrives ut av denne koden?

[18. 19. 20. 21. 22. 23. 24. 25.]

```
import numpy as np

a = np.arange(1,5,0.5)
c = a.size
b = (a+c)*2
print(b)
```

Oppgave 3 - Programmering

Førerkortprikker og botbetaling

I denne eksamenen skal du lage et system for registrering av prikker som norske borgere får når de begår trafikkforseelser. Du skal lage et sett funksjoner, som oppfyller en del krav til systemet, disse blir beskrevet under hver oppgave. Dette er et oversiktsbilde av systemet, så følger en beskrivelse av datastrukturen som skal lages og modulen som allerede eksisterer. Merk for øvrig at dette oppgavesettet kun er inspirert av det faktisk prikkbelastningssystemet som finnes i Norge.

Modulen nameregister.py

Alle personer som registreres i systemet må registreres med fødselsnummer og navn (fornavn etternavn). Begge disse er av type strenger. Modulen nameregister inneholder tre funksjoner som hjelper deg med å registrere denne koblingen. Modulen ligger i den samme folderen som koden du skal skrive, du skal altså bruke disse i koden din. De tre funksjonene er

register_name(...):

Denne brukes til å legge til nye fødselsnummer og knytte dem til et navn.

find_name(...):

Denne funksjonen brukes til å finne et navn gitt at en oppgir et fødselsnummer (11 siffer som streng).

Begge disse funksjonene kalles med en variabel people. Denne kan du forvente å ha etablert i koden du skriver i alle funksjonene på denne eksamenen. For en ytterligere beskrivelse av funksjonene, inkludert parametere, refererer vi til beskrivelsen av modulen som ligger i bunnen av funksjonsbeskrivelsen til venstre for oppgavesettet. Du skal altså ikke gå rett inn og endre på people, dette er kun tillatt gjennom funksjonene i modulen.

days_in_month(...):

Denne funksjonen tar inn et heltall mellom 1 og 12, og returnerer hvor mange dager det er i denne måneden. Funksjonen ignorerer skuddår. Funksjonen krasjer for andre innverdier enn tallene 1-12! Eksempel: days_in_month(1) returnerer 31

Oppgavens struktur

En funksjon du skal lage i en oppgave kan med hell benytte seg av funksjoner som allerede er beskrevet i tidligere oppgaver. Du står fritt til å bruke tidligere funksjoner slik de står beskrevet selv om du ikke har klart å løse dem - bruk funksjonene slik de er spesifisert å virke. Det vil aldri være tilfelle at du skal bruke funksjoner som blir gitt som oppgaver i senere oppgaver. Oppgavene kan løses i nedadgående rekkefølge. Noen oppgaver er vanskeligere enn andre. Det er lurt å bruke litt tid til å lese igjennom alle først, for å danne seg et inntrykk av hvordan en skal angripe oppgaven.

Datastruktur

Datastrukturen som du skal utvikle består denne av en dictionary som kalles register. Nøkkelen er et fødselsnummer (11 siffer lang streng). Verdien knyttet til hver nøkkel er prikkene som hver fører har fått så langt. Prikkene er lagret i form av en liste, der nye forseelser legges inn til slutt i listen. Eksempelvis vil et register med to førere som begge har fått noen prikker se slik ut:

register = {'12046712125':[2,2,3],'23099045653':[3]}

Det er ikke nødvendig å tenke på strukturen til det som finnes i modulen - her trenger en bare å bruke modulens funksjoner på korrekt måte. Du kan forvente at det eksisterer en variabel people som brukes når en kaller opp funksjonene her.

Alle oppgavene krever at du skal skrive en funksjon. Når dere ser beskrivelsen (...) etter en funksjon betyr det at en skal fylle inn antall parametre og navnet på dem slik det står beskrevet i oppgavene.

3a - Datosjekk

Nye førere må registreres (mer om det senere), og da må en legge inn folks fødselsnummer (streng med 11 siffer). Vi ønsker ikke å legge inn verdier som er feil i systemet. Derfor må det kontrolleres at datoer som legges inn i alle fall er reelle.
Skriv funksjonen **check_date**

- Funksjonen skal ta inn en streng som skal verifiseres som en reell dato. Strengen er hele fødselsnummeret til føreren, der de seks første dekker fødselsdatoen. Disse er på formatet ddmmyy - 12 mars 1979 blir da '120379'. Litt informasjon og krav:
- Alder: år personen er født (kun de to siste sifrene). Du kan se bort fra alder på personene, altså ingen sjekk om på om de gamle nok til å ha førerkort.
- Måned: tallet må være mellom 01 og 12, og alltid med to siffer (mars blir '03')
- Dag: dag passer inn med måneden brukt.
- Det finnes allerede en funksjon days_in_month(month). Denne ligger i modulen nameregister.py i den samme mappen som programmet ditt. Denne må du gjerne bruke, og du finner en beskrivelse av den nederst på funksjonsarket.
- Du kan også se bort fra alt som har med skuddår å gjøre.
- Hvis datoen er reell skal funksjonen returnere True, ellers skal den returnere False.

Eksempel:

```
>>> print(check_date('29127311245')) # Skriver ut True mens
>>> print(check_date('31117532456')) # Skriver ut False
```

Løsningsforslag:

```
import nameregister as nr

def check_date(dato): # Dato inn: 13037954321 -> True
    date = dato[:6]
    year = date[4:]
    month = int(date[2:4])
          # gjør om til tall siden vi skal sammenlikne senere.
    day = int(date[:2])
    if 0 < month < 13:
          # Måneden er rett, nå må kan vi returnere om dagen er
          # rett:
        return nr.days_in_month(int(month)) >= day
    return False
```

3b - Registrer ny fører

Når en ny person begår en trafikkforseelse må denne registreres. Her er det to ulike systemer dette skal registreres i - people og register.
Skriv funksjonen **register_person**
Funksjonen tar fire parametre:

- people: En dictionary som gir kobling mellom fødselsnummer og navn.
- register: En dictionary som har fødselsnummer (streng) som nøkkel og en liste over prikker som verdi.
- fnr: En streng som inneholder fødselsnummeret til den som skal registreres. Du kan forvente at datoen her er korrekt, og at det ikke finnes noen med dette fødselsnummeret fra før.
- name: En streng som inneholder navnet til den som skal registreres.

Funksjonen skal gjøre følgende:

- I people skal du legge til et nytt innslag med fnr og navn. register_name i modulen nameregister bør være til hjelp.
- I register skal det registreres et nytt innslag med nøkkel fnr, verdien skal være en tom liste.
- Variablene people og register kan du forutsette at du har tilgang på allerede - du trenger altså ikke lage dem. Dette gjelder også seinere oppgaver.

Eksempel:

```
>>> print(people) # Tom dictionary
>>> {}
>>> print(register) # Tom dictionary
>>> {}
>>> register_person(register,'11010154321','Terje Rydland')
>>> register_person(register,'23056644521','Børge Haugset')
>>> print(people)
{'11010154321:'Terje Rydland','23056644521':'Børge Haugset'}
>>> print(register)
{'11010154321':[],'23056644521':[]}
```

```
def register_person(register, people,fnr,name):
    nr.register_name(people,fnr,name)
    register[fnr] = []
    return people, register # unødvendig for dictionaries
```

3c - Telle prikker

Personer får registrert prikker i en dictionary kalt ***register***. Denne har som nøkkel et ***personnummer*** (streng med 11 siffer). Verdien er en liste som inneholder alle prikkene personen har fått, den første først.
Skriv funksjonen **count_dots**

- Du kan forvente at du har tilgang til variabelen register
- Funksjonen skal ha to parametre:
    - Den første er register (dictionary), som er nevnt over.
    - Den andre er fnr (streng, 11 siffer), fødselsnummeret til føreren man skal finne antall prikker til.
- Funksjonen skal returnere summen av prikker (som heltall) denne personen har fått.
- Du kan forvente at fødselsnummeret som oppgis finnes i register.

```
>>> # They have respectively 8 and 3 dots
>>> register = {'11010154321':[3,2,3],'23056644521':[3]}
>>> count_dots(register,'11010154321')
8
```

```
def count_dots(register, fnr):
    return sum(register[fnr])
```

3d - Legge til prikker

Prikker legges inn som verdi i dictionary register, knyttet til nøkkelen fødselsnummer (streng med 11 siffer). Prikkene er lagret i listeform, og nye prikker registreres på slutten av den eksisterende listen. Listen består av heltall, og kan se slik ut: [2, 2, 3]. Når nye prikker har blitt lagt til må en sjekke om totalt antall prikker er 10 eller mer. Hvis de er det skal det skrives ut en beskjed om at føreren har gått over grensen, og førerkortet skal beslaglegges. Her holder det å skrive ut en beskjed til brukeren som legger inn prikkene.
Skriv funksjonen **add_dots**

- Funksjonen skal ha tre parametre
    - register - dictionary som inneholder fødselsnummer og prikker for alle, som før.
    - fnr - fødselsnummeret til den som har kjørt i grøfta. Du kan forvente at dette fødselsnummeret allerede er registrert.
    - dots - antallet prikker som skal legges til i slutten av denne førerens liste.
- Hvis antallet prikker for føreren etter innlegging av nye prikker er 10 eller mer, skal det skrives ut en beskjed. Se eksempelet under. Husk funksjoner som er laget før selv om du ikke har skrevet dem selv.
- Du kan forvente å ha tilgang til variablene people og register.

Eksempel:

```
>>> print(register)
{'11010154321':[3,2,3],'23056644521':[3]}
>>> add_dots(register,'11010154321',3)
The driver has more than 10 dots, confiscate!
>>> add_dots(register,'23056644521',3)
>>> # Nothing is printed, as the total is less than ten.
```

```
def add_dots(register, fnr,dots):
    register[fnr] = register[fnr] + [dots]
    new_dots = count_dots(register,fnr)
    if new_dots >= 10:
        print(f'The driver has more than 10 dots, \
                confiscate!')
```

3e - Manuell innlegging av prikker

Funksjonene som er skrevet til nå har alle basert seg på at informasjon som skal legges inn allerede er definert. Nå har vi kommet til delen der brukeren av systemet må skrive inn ting.

Skriv funksjonen **manual_registration**

- Funksjonen har to parametre - ***register*** og ***people***. De er like som i oppgavene før.
- Brukeren skal først skrive inn et fødselsnummer (streng med 11 siffer).
- Det må sjekkes om fødselsdatoen i fødselsnummeret er en reell dato. Hvis ikke - avbryt funksjonen. Fødselsdatoen er de seks første sifrene i fødselsnummeret.
- Hvis dette fødselsnummeret ikke allerede er registrert må denne nye personen registreres i både ***people*** og ***register***, i tråd med funksjonsbeskrivelsene gitt i tidligere oppgaver. Husk å spørre om navn!
- Til slutt skal funksjonen spørre om hvor mange prikker denne føreren har fått, og registrere dem.

```
>>> # Existing user:
>>> manual_registration(register,people)
Personal ID number: 23056644521
Dots to register: 2
>>> # Invalid date of birth
>>> manual_registration(register,people)
Personal ID number: 12345678909
The date is invalid
>>> # New ID number
>>> manual_registration(register,people)
Personal ID number: 30118866154
Not registered before. Name: Alf Inge Wang
Dots to register: 1
```

Løsningsforslag:

```
def manual_registration(register, people):
    fnr = input('Fødselsnummer: ')
    if not check_date(fnr):
        print(f'The date is invalid.')
    elif not nr.find_name(people,fnr):
        name = input('Not registered before. Name: ')
        registrer_person(register,people,fnr,name)
    dots = int(input('Dots to register: '))
    add_dots(register,fnr,dots)
```

3f - Betale seg ut av problemer

Det er alltid mulig å betale seg ut av problemer i dette systemet. En fører har mulighet til å betale for å redusere antallet prikker som er registrert på seg.
Skriv funksjonen **pay**

Funksjonen **pay** har en parameter:

- register: Registeret over fødselsnummer og nåværende prikker.

Den fungerer på følgende måte:

- Brukeren skal bli spurt om å skrive inn fødselsnummeret og summen føreren skal betale.
- Funksjonen skal legge inn negative prikker. 1 prikk per 10 000 kroner. Hvis en fører vil betale 35 000 kroner skal tallet -3 legges til på slutten av listen, som i eksempelet under. Husk tidligere funksjoner.
- Hvis føreren ikke er registrert skal meldingen 'Fører ikke registrert' skrives ut og funksjonen avsluttes.

Eksempel:

```
>>> register = {'11010154321':[3,2,3,2],'23056644521':[3]}
>>> pay(register)
Personal ID number: 11010154321
How much will the driver pay? 35000
>>> print(register['11010154321'])
[3,2,3,2,-3]
```

Løsningsforslag

```
def pay(register):
    fnr = input('Personal ID number: ')
    avlat = int(input('How much will the driver pay? '))
    reduction = -(avlat//10000)
    add_dots(register,fnr,reduction)
```

Oppgave 3g - Lagring til fil

Vi ønsker å lagre informasjon om førerne som er registrert på fil.
Skriv funksjonen **save_to_file**

- Funksjonen tar to parametre, register og people
- Filen det skal skrives til skal hete 'dots.txt'.
- For hver av de registrerte førerne skal det lages en linje der følgende streng står (uten ' foran og bak): '11010154321 (Terje Rydland) har 7 prikker.'

Eksempel:

```
>>> register = {'11010154321':[3,2,3,2],'23056644521:[3]}
>>> people = {'11010154321':'Terje
Rydland,'23056644521:'Børge Haugset'}
>>> save_to_file(register,people)
```

Etter at koden over har kjørt, vil dots.txt inneholde følgende:

```
11010154321 (Terje Rydland) har 10 prikker.
23056644521 (Børge Haugset) har 3 prikker.
```

Løsningsforslag:

```
def save_to_file(register, people):
     with open('dots.txt','w') as f:
          for fnr in register:
               f.write(f'{fnr} ({nr.find_name(people, fnr)})'+\
                    f'har {count_dots(register, fnr)} dots.\n')

# Alternativt
def save_to_file(register, people):
     f = open('dots.txt','w')
     for fnr in register:
          f.write(f'{fnr} ({nr.find_name(people, fnr)})'+\
               f'har {count_dots(register, fnr)} dots.\n')
     f.close()
```