Løsningsforslag, TDT4110, 10/12 kl 0900

Oppgave 1 - Teori

Hva er Alan Turing kjent for?

Han utformet det matematiske grunnlaget for dagens datamaskiner.

I et moderne minne består hver lokasjon av

1 byte

Kontrollenheten har to registre. Det ene er programtelleren. Hva heter det andre registeret?

Instruction register

Hvilket utsagn omkring kompilering og tolking er korrekt

Ved tolking oversettes kodelinjene en for en og utføres med en gang

En device driver er

Spesialisert programvare for input/output, slik at utstyr kan kommunisere med resten av systemet.

Hva sier Nyquist-regelen for sampling.

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.

Hvilket binært tall representeres av det hexadesimale tallet 39A

0011 1001 1010

Hvor mange symboler kan representeres med 6 bit

64

Digital informasjon er

diskret

Hva blir -14 i 2ers komplement representert med 8 bit

11110010

Hva er misrepresentation

Å oppgi feilaktige opplysninger om et produkt eller tjeneste eller levere produkter/tjenester som er falsk eller av dårlig kvalitet.

Hva går buffer overflow ut på

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.

Hva er password breaking

Automatiserte systemer laget for å knekke passord og dekrypteringsnøkler for å få uautorisert aksess til en nettressurs.

Hva er ikke sant om brannmurer (firewalls)?

En sikkerhetsteknologi som tjener dataintegritet.

Hvorfor oppstår det ofte forsinkelser ved bruk av *VPN*

VPN resulterer gjerne i at datapakker ofte må gå fram og tilbake over internett flere ganger mellom en bruker og det lokale nettverket han er koblet opp mot.

Hva er IKKE sant om Cloud computing

Cloud services gir som regel bedre ytelser på nettjenester for en bedrift, men er som regel dyrere drift enn å ha egne tjenere (servere).

Hva er IKKE sant om kanalkoding (channel coding)?

Kanalkoding er å kryptere meldinger fra avsender til mottaker slik at andre ikke kan lese innholdet på meldingen.

Hva er det som muliggjør internettets illusjon av et enkeltstående, sømløst kommunikasjonssystem bestående av mange datamaskiner?

TCP/IP

Hva er replay error i nettverkssammenheng

At en forsinket pakke fra tidligere sesjon blir akseptert i senere sesjon, og at korrekt pakke dermed blir avvist som duplikat.

Oppgave 2 - Kodeforståelse

2a

Hva skrives ut av denne koden?

00100101

```
def myst2(a):
    b = [] + ['0']*len(a)
    for i in range(len(a)):
        if a[i]:
            b[i] = '1'
    return b

def myst(streng):
    a = list(streng)
    for i in range(len(a)):
        a[i] = int(a[i])
        a[i] = not a[i]
    return myst2(a)

print(''.join(myst('11011010')))
```

2b

Hva skrives ut av denne koden?

[82, 19, 65, 17, 99]

```
def myst9(minListe):
    nyListe = []
    while len(minListe) > 0:
        ind = 0
        minste = minListe[ind]
        for i in range(len(minListe)):
            if minListe[i] < minste:
                ind = i
        nyListe.append(minListe.pop(ind))
    return nyListe

print(myst9([99,17,65,19,82]))
```

2c

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

a=0, b=6

```
def myst(s,x,y):
    s1 = ''
    for i in range(x,len(s),y):
        s1+= s[i]
    return s1

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

2d

Hva skrives ut når denne koden kjøres?

[2,3,4]

```
def myst(noe):
    noe2 = set(noe[0])
    for i in range(1,len(noe)):
        noe2 = noe2.intersection(set(noe[i]))
    return list(noe2)

print(myst([[1,2,2,3,3,4,5],[1,2,2,2,4,3,3],[4,5,2,2,3,3,3,8],[2,2,3,3,4,5,7]]))
```

2e

Hva skrives ut når denne koden kjøres?

10100110
(Koden lager 2ers komplement)

```
def myst(noe):
    a = '01'
    verdi = noe[-1]
    f = noe[-1] == a[1]
    for i in noe[-2::-1]:
        if not f:
            verdi = i+verdi
            if i == a[1]:
                f = True
        else:
            if i == a[1]:
                verdi = a[0]+verdi
            else:
                verdi = a[1] + verdi
    return verdi


print(myst('01011010'))
```

2f

Hva skrives ut når denne koden kjøres?

120

```
def myst(a,b):
    c,d = 0,0
    for i in a[::-1]:
        d += int(i)*b**c
        c += 1
    return d

print(myst('11110',3))
```

2g

Hva skrives ut når denne koden kjøres?

-3

```
def myst(x):
    a,b,c = 0,0,0
    while a <= x:
        c += 1
        b = a
        if c % 2 == 0:
            a -= c**2
        else:
            a += c**2
    return b

print(myst(5))
```

2h

Hva skrives ut når denne koden kjøres?

None

```
def myst(a,b):
    if a > b:
        c = a
    else:
        c = b
    for i in range(2,c+1):
        if a%i == 0 and b%i == 0:
            return i

print(myst(30,7))
```

2i

Hva skrives ut når denne koden kjøres?

7/2

```
def myst2(a,b):
    while b != 0:
        c = b
        b = a % b
        a = c
    return a

def myst(a,b):
    c = myst2(a,b)
    a = a//c
    b = b//c
    return a,b

a,b = myst(42,12)
print(f'{a}/{b}',sep='')
```

2j

Hva blir b etter at denne kodebiten er kjørt?

[[20 22] [24 26] [28 30] [32 34]]

```
import numpy as np

a = np.array([[2,3,4,5],[6,7,8,9]])
c = a.size
b = (a+c)*2
b=b.reshape((4,2))
print(b)
```

Oppgave 3 - Programmering

3a - Registrering av fiskebåt

Du tar imot fisk fra kjente og nye fiskebåter. Fiskeren registrerer en ny båt gjennom å taste inn båtens nummer i henhold til fiskeriregisteret. I bakgrunnen kaller datasystemet opp en funksjon kalt check_registration med regnummer som parameter. Funksjonen returnerer True eller False alt etter om registreringsnummeret følger retningslinjene i registeret. Reglene er som følger: Nummer i fiskeriregisteret har formen bokstav-tall-bokstav, for eksempel N64Ø («Solbris»). Første bokstav tilsvarer fylkesbokstaven som ble brukt på bilskilt fram til omkring 1970. Tallene gis fortløpende til fiskere i hver enkelt kommune, og kan derfor ha enten et eller to sifre. Til slutt står en eller to bokstaver som angir kommune. Eksempelet N64Ø som er oppgitt viser at dette var fiskefartøy nummer 64 i Øksnes kommune i Nordland. Vi gjentar: <1 bokstav><1-2 siffer><1-2 bokstaver>.
Skriv funksjonen check_registration som skal ta imot et registreringsnummer og returnere om det er skrevet i henhold til retningslinjene i fiskeriregisteret.
Eksempel:

```
>>> print(check_registration('N64Ø')) # Skriver ut True mens
>>> print(check_registration('NN4Ø')) # Skriver ut False
```

Eksemplene over er kun eksempler - man skal ikke 'hardkode' sjekk mot akkurat disse.
Følgende registreringsnummer skal for eksempel også returnere True: N4A, B46MC, P53V, Z4HB.
Følgende registreringsnummer skal returnere False: NR5H, P557J, 672NH.

```
def check_registration(regnummer):
    if not (regnummer[0].isalpha() and regnummer[1].isdigit() \
          and regnummer[-1].isalpha()):
        return False      # Hvis første ikke er en bokstav, andre
                          # ikke er et tall og den siste ikke er en
                          # bokstav kan ikke regnummeret være rett
    rest = regnummer[2:-1] # Resten igjen
    if len(rest) == 0:  # Det må være 0, 1 eller 2 tegn i rest
        return True
    elif len(rest) == 1:
        return rest.isdigit() or rest.isalpha()
    elif len(rest) == 2:
        return rest[0].isdigit() and rest[1].isalpha()
    else:
        return False
```

3b - Hvor mye av en viss fisk

Hver båt har lagret fisken i en todimensjonell liste. Det kan være fint å finne ut hvor mye av en enkelt fiskeart som er lagret. Her kan vi ta for gitt at lagret fisk for en gitt fisker er tilgjengelig gjennom parameteren store, mens fisketypen er en streng gitt i parameteren kind. Hvis det ikke finnes noe fisk av denne typen skal funksjonen returnere tallet 0.
Skriv funksjonen **fish_amount** som tar parametrene **store** og **kind**.
Eksempel:

```
lager = [['torsk', 200],['sei',100]]
>>> print(fish_amount(lager, 'torsk')) # Skriver ut 200
>>> print(fish_amount(lager, 'månefisk')) # Skriver ut 0
```

```
def fish_amount(store, kind):
    for type in store:
        if type[0] == kind:
            return type[1]
    return 0
```

3c - Legg til fisk

Lageret skal oppdateres når det kommer inn mer fisk. Dette oppgis som en liste med fiskeslag og antall kilo: ['torsk', 200]. Lag et nytt element hvis fisketypen ikke allerede finnes i lageret. Skriv funksjonen **add_fish** som tar store og en liste som inputparameter og returnerer det oppdaterte lageret.
Eksempel:

```
>>> print(store)
[['torsk', 200],['sei',100]]
>>> add_fish(store,['torsk',100]) # Already in store
>>> add_fish(store,['hyse',70]) # Not in store
>>> print(store)
[['torsk', 300],['sei',100],['hyse',70]]
```

```
def add_fish(store, fangst):
    funnet = False # Eksisterer fisketypen fra før
    for type in store:
        if fangst[0] == type[0]:
            type[1] = type[1] + fangst[1]
            funnet = True
    if not funnet:
        store.append(fangst)
    return store # strengt tatt unødvendig da store er
                 # muterbar
```

3d - Lagre masse fisk

En fisker kan fange mange typer fisk på en gang - og ønsker dermed å rapportere inn dette. Flere typer fisk blir rapportert inn som en todimensjonal liste, på samme format som store: [['type', mengde], ['type', mengde], ['type', mengde]]
Skriv funksjonen **add_much_fish** som tar store og en 2D-liste som inputparametre.
Eksempel:

```
>>> print(store)
[['torsk', 300],['sei',100],['hyse',70]]
>>> add_much_fish(store,[['kveite',120],['torsk',200]])
>>> print(store)
[['torsk', 500],['sei',100],['hyse',70],['kveite',120]]
```

```
# Her er det bare snakk om å kalle funksjonen add_fish
# det antall ganger det er elementer i total_fangst

def add_much_fish(store, total_fangst):
    for enkelt_fangst in total_fangst:
        add_fish(store, enkelt_fangst)
```

3e - Fjerne fisk

Fisk fjernes fra listen av to grunner - enten har den blitt for gammel, eller så er den solgt (mer om det senere). Hva og hvor mye som skal fjernes sendes med som parameter, på formatet ['type', mengde].
Lag funksjonen **remove_fish** som har parametrene ***store*** og ***remove***.
Funksjonen skal fjerne den gitte mengden fisk fra lageret (store). Dersom det skulle gjøres et forsøk på å fjerne mer fra lageret enn det faktisk finnes, så skal det gis beskjed om dette. I så fall skal ingen fisk fjernes når funksjonen kalles. Du finner et eksempel på dette ved spørring etter kveite under.

```
>>> print(store)
[['torsk', 500],['sei',100],['hyse',70],['kveite',120]]
>>> remove_fish(store,['torsk',300]
[['torsk', 200],['sei',100],['hyse',70],['kveite',120]]
>>> remove_fish(store,['kveite',200])
There's not enough kveite left
>>> print(store)
[['torsk', 200],['sei',100],['hyse',70],['kveite',120]]
```

Løsningsforslag:

```
def remove_fish(store, remove):
    for fish in store:
        if fish[0] == remove[0]:
            if fish[1] >= remove[1]:
                fish[1] -= remove[1]
            else:
                print(f"There's not enough {fish[0]} \
                        left")
    return store # Denne er ikke nødvendig da store er
                 # mutable og dermed oppdateres
                 # automatisk
```

3f - Lese inn fisk fra fil

Fiskebåtene er aktive på natta. Under opptelling rapporterer de inn fangstmengden av ulike fiskeslag gjennom nettsiden din, dette lagres rett i en fil. Når du kommer på kontoret kjører du funksjonen read_fish_from_file. Denne funksjonen leser innholdet i filen fishy.txt og oppdaterer et register med båter og hvilke fisketyper og mengder de har tilgjengelig. Filen består av et sett med linjer på formen båtnavn:fisketype:mengde. (Andre og sjette innslag her er feil)

N64Ø:torsk:343
N434:torsk:200
Z4F:torsk:120
N64Ø:torsk:200
N64Ø:hyse:110
NB4A:kveite:100
Z4F:hyse:120
N64Ø:kveite:300

Skriv funksjonen **read_fish_from_file**. Funksjonen tar inn en dictionary register - denne er beskrevet i første del av oppgaveteksten. Dersom det ikke eksisterer en båt med båtnavnet i registeret fra før, og båtnavnet stemmer med navnesjekken i oppgave a skal det opprettes en ny nøkkel på denne og fangsten som er rapportert skal legges inn. Dersom båtnavnet mot formodning ikke er korrekt bygget opp skal det ikke legges til i registeret, men funksjonen skal i stedet skrive ut:
'Båtkode' var ugyldig - hopper over fangstrapportering.

Eksempel:

```
>>> register = {}
>>> read_fish_from_file(register)
N434 var ugyldig - hopper over fangstrapportering
NB4A var ugyldig - hopper over fangstrapportering
>>> print(register)
{'N64Ø':[['torsk',543],['hyse',110],['kveite',300]],
'Z4F':[['torsk',120],['hyse',120]]}
```

Løsningsforslag

```
def read_fish_from_file(register):
    f = open('fishy.txt','r')
    data = f.readlines()
    f.close()
    for linje in data:
        navn,art,mengde = linje.strip().split(':')
        if not check_registration(navn):
            print(f'{navn} var ugyldig - hopper over \
                 fangstrapportering.')
        else:
            if navn not in register:
                register[navn] = []
            add_fish(register[navn],[art,int(mengde)])
    return register # Denne er ikke nødvendig da register
                    # er mutable og dermed oppdateres
                    # automatisk
```

Oppgave 3g - Send kjøpere til fiskerne som har rett type

Gjennom nettsiden din sender kjøpere deg forespørsler om de kan kjøpe en mengde fisk av ulik type og mengde, men bare en type av gangen.
Funksjonen **handle_customer** med parametrene ***register*** og ***need*** håndterer dette. Kundene skal få vite hvilke båter de må kontakte, og hvor mye fisk av ulikt slag de kan kjøpe fra hver av båtene. De skal i tillegg få beskjed om hvor mye dette koster, og få vite det dersom det ikke er nok fisk av typen til å dekke behovet. Den skal også beregne din inntekt av videreformidlingen.
Skriv funksjonen **handle_customer** som beskrevet.

Variabelen *register* inneholder den fisken som er tilgjengelig hos alle fiskeselgerne akkurat nå, mens *need* er fiskearten og mengden som kunden vil ha. Formatet på need er en liste ['type', mengde]. Dette behovet skal så fylles av én eller flere fiskebåter, hvis mulig. Du trenger ikke tenke på rettferdig fordeling av salg mellom fiskebåtene. Hvis det ikke er mulig å samle nok fisk til å dekke behovet til kunden skal det skrives ut en beskjed, se eksempelet under. Dersom behovet er dekket skal tilsvarende mengder av fisk som kunden skal ha fra hver båt fjernes fra registeret til båtene.
Hvis et behov ikke kan dekkes skal ingen fisk fjernes. Hvis et salg lykkes skal du ha *betalt fra kunden*. Du tar en fast pris på ti kroner per kilo for all fisk som omsettes, og femti kroner for hver båt som det skal handles fra. Det skal skrives ut til kunden hvor mye handelen koster samtidig som de får beskjed om hvilke båter de kan handle fra.

Eksempler:

```
>>> register = {'N64Ø':[['torsk',543],['hyse',110],
['kveite',300]], 'Z4F':[['torsk',120],['hyse',120]]}
>>> handle_customer(register,['torsk',300])
The following boats have torsk: [['N64Ø',300]]. The total
cost is 3050 kroner.
>>> handle_customer(register,['torsk',300])
The following boats have torsk: [['N64Ø',300],
['Z4F',57]]. The total cost is 3100 kroner.
>>> handle_customer(register,['kveite',310])
There's not enough kveite left to cover your requirement
>>> print(register)
{'N64Ø':[['torsk',0],['hyse',110],['kveite',300]], 'Z4F':
[['torsk',63],['hyse',120]]}
```

Løsningsforslag:

```
def handle_customer(register, need):
    betjent = True
    # Settes til false underveis dersom vi ikke kan
    # betjene behov
    resultat = []
    # Listen som skal sendes til kunde, og brukes til
    # sletting hvis alt finnes.
    art, mengde = need
    for regnr, store in register.items():
        regnr_har = fish_amount(store, art)
        if regnr_har >= mengde:
            resultat.append([regnr,mengde])
            break # Vi er ferdige
        elif regnr_har > 0:
            resultat.append([regnr,regnr_har])
            mengde -= regnr_har
             # mengde reduseres gradvis etterhvert som
             # fiskere fyller behovet.
        else: # kunden trenger mer enn vi har
            betjent = False
    if betjent:
        print(f'Følgende båter har {art}: {resultat}. \
             Det koster {(need[1]*10)+50*len(resultat)} \
             kroner.')
        for regnr, mengde_per_fisker in resultat:
            remove_fish(register[regnr],\
                        [art,mengde_per_fisker])
    else:
        print(f'Det er ikke nok {art} til å \
                  tilfredsstille kravet ditt.')
```