NTNU
Kunnskap for en bedre verden

KANDIDAT
10520

PRØVE
TDT4109 1 Informasjonsteknologi, grunnkurs

Emnekode	TDT4109
Vurderingsform	Hjemmeeksamen
Starttid	02.12.2020 08:00
Sluttid	02.12.2020 12:00
Sensurfrist	06.01.2021 22:59
PDF opprettet	27.01.2021 16:01

Front Page

Oppgave	Tittel	Oppgavetype
i	Forside	Dokument

Theory Questions

Oppgave	Tittel	Oppgavetype
i	Oppgave 1 (25%)	Dokument
1	Oppgave 1.1 (5%)	Langsvar
2	Oppgave 1.2 (5%)	Langsvar
3	Oppgave 1.3 (5%)	Langsvar
4	Oppgave 1.4 (5%)	Langsvar
5	Oppgave 1.5 (5%)	Langsvar

Programming: Diverse problems

Oppgave	Tittel	Oppgavetype
i	Oppgave 2 (25%)	Dokument
6	Oppgave 2.1 (5%)	Programmering
7	Oppgave 2.2 (5%)	Programmering
8	Oppgave 2.3 (5%)	Programmering
9	Oppgave 2.4 (10%)	Programmering

Programming: PokerWar

Oppgave	Tittel	Oppgavetype
i	Oppgave 3 (50%)	Dokument
10	Oppgave 3.1 (5%)	Programmering
11	Oppgave 3.2 (10%)	Programmering
12	Oppgave 3.3 (10%)	Programmering

13	Oppgave 3.4 (10%)	Programmering
14	Oppgave 3.5 (5%)	Programmering
15	Oppgave 3.6 (10%)	Programmering

1 Oppgave 1.1 (5%)

Forklar kort, komplett og konsist hvordan kjøring av instruksjoner foregår i en datamaskin og forklar hvilke oppgaver minnet (memory), kontrollenheten (control unit) og aritmetisk logisk enhet (ALU) har.

Skriv ditt svar her

Man har tre faser ved kjøring av instruksjoner: fetch, decode og execute.
Fetch fase: Her henter man adressen til neste instruksjon fra instruksjonsadresse-registeret og flytter neste instruksjon fra RAM til instruksjonsregisteret.

Decode fasen: Først dekoder man bitsene i instruksjonsregisteret, så dekoder man instruksjonen og til slutt henter man minneadressen for data som skal hentes fra RAM

Execute fasen: Først utfører man instruksjonen. Så leser man data fra RAM til register og til slutt skriver man data fra register til RAM.

Memory: Brukt for rask lagring og rask tilgang av data for cpu. Mye raskere enn sekundærminne, men krever strøm for å lagre dataen.

Control unit: Brukes for å fortelle de ulike kompontene hvordan de skal svare på ulike signaler.

ALU: Denne blir brukt for aritmetikk, addisjon subtraksjon.

2 Oppgave 1.2 (5%)

Forklar kort, komplett og konsist hvordan 24 bits kan representere et punkt i et bilde med ca. 16 millioner fargenyanser:

Skriv ditt svar her

(r,g,b) (8 bits, 8, bits, 8bits)
Siden 2^24 er tilnærmet 16 millioner kan man manipulere hver fargekanal til at man får ca. 16 millioner fargenyanser i hvert piksel. Siden hver piksel består av tre fargekanaler kan man gi et digitalt signal med 8 bits dypbe til hver fargekanal.

3 # Oppgave 1.3 (5%)

Forklar kort, komplett og konsist grunner til at man kan få feil i overføring av data over et kommunikasjonsmedium:

Skriv ditt svar her

Man har hovedsaklig tre typer grunner: interferens, forvrengning og demping.

Demping er når signalet er for svakt til å kunne få et klart signal. Dette kan komme av store overføringsdistanser.

Forvrenginger når signlaet blir endret, altså faseforskjøvet eller endret amplitude. Dette kan komme av brudd i kabler og ødeleggelser av fysiske systemer.

Interferens kommer av andre signaler som enten forsterker eller destruerer signalet. Dette fororsakes av annen støy i kommunikasjonsmediumet.

4 # Oppgave 1.4 (5%)

Sammenlikn kort, komplett og konsist fordeler og ulemper med metodene Single Parity Checking (SPC), Error Correction With Row and Column (RAC) Parity, 16-Bit Checksum, og Cyclic Redundancy Codes (CRCs):

Skriv ditt svar her

SPC: Veldig raskt, siden man kun legger til en bit. Men det det er lite sikkert, og kan lett overse feil.

RAC: Sikrere enn SPC, men kreves mer maskinvare for å sjekke.

16-bit checksum:

CRCs: Fungerer uansett lengde på data, og har meget bra feildeteksjon. I tilleg kan det utføres veldig raskt med maskkinvare.

5

Oppgave 1.5 (5%)

Sammenlikn kort, komplett og konsist følgende sikkerhetsproblem iht. hvor store skade et enkelt angrep gjør: Phishing, Scams, Denial of Service, Tap av data (Loss of Data).

Skriv ditt svar her

Phising er når man opptrer som en kjent nettside for å få tak i personlig informasjon. Her blir personlig informasjon, som passord, kontoinformasjon eller private mailer samlet inn. Dette kan bli lekket, eller så kan det bli krevet løsepenger for at det ikke skal bli lekket. Så er slikt vellykket angrep kan ha store økonomiske tap og kan ødelegge firmaer.

Scams er når man lurer brukere til å investere eller bruke penger. Her er det kun økonomiske konsekvenser. Desverre er det oftest slik at enkeltpersoner blir utsatt for scams og taper mye penger.

Denial of Service er når man blokkerer tilgang til en tjeneste. Hvis inntrengere greier å holde angrepet oppe over tid, kan selskaper tape masse penger, og muligens kunder som vil inn på siden. Dette vil da kun gi kortsiktig økonomisk skade.

Tap av data er når man mister åndsverk eller annen viktig informasjon. Hvis man mister åndsverk eller verdifull informasjon vil dette være et langsiktig problem. Her kan bruker-passord bli delt, firma-hemmeligheter kan bli lekket og produkter kan bli kopiert. Et slikt angrep gjøre veldig mye skade.

6

Oppgave 2.1 (5%)

Lag funksjonen ***check_across_div*** som har to input-parametere ***num1*** og ***num2***, som begge er heltall.
Funksjonen skal sjekke om tverrsummen av ***num1*** er delelig med ***num2***.
Tverrsummen av et tall finner man ved at man summerer sifferne i tallet:
Tverrsummen av 123 = 1 + 2 + 3 = 6, og tverrsummen av 255 = 2 + 5 + 5 = 12
Funksjonen skal returnere ***True*** hvis tverrsummen av ***num1*** er delelig med ***num2***. Hvis ikke, skal ***False*** returneres.

Eksempel på kjøring av kode:

Shell

```
>>> check_across_div(123,3)
True
>>> check_across_div(123,4)
False
>>> check_across_div(235,4)
False
>>> check_across_div(235,5)
True
>>>
```

Skriv ditt svar her

```
def check_across_div(num1, num2):
    tverrSum = 0
    for letter in str(num1):
        tverrSum += int(letter)

    if tverrSum % num2 == 0:
        return True
    else:
        return False
```

7

Oppgave 2.2 (5%)

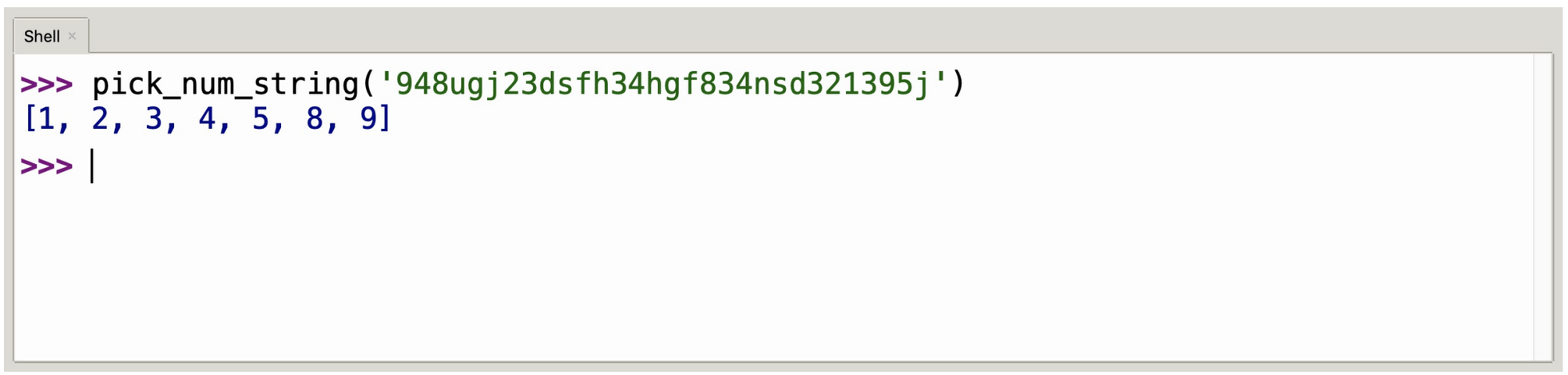
Lag funksjonen ***pick_num_string*** som har en input-parameter ***txt*** som er en tekststreng som kan bestå av både bokstaver og tall.

Funksjonen skal returnere alle tallene som ble funnet i tekststrengen som en liste i stigende rekkefølge. Lista skal ikke ha duplikater av samme tall hvis man finner tallet flere ganger i ***txt***.

Funksjonen kan testes med følgende argument:

txt='948ugj23dsfh34hgf834nsd321395j'

Eksempel på kjøring av kode:



Skriv ditt svar her

```
def pick_num_string(txt):
    tallListe = []
    for letter in txt:
        if letter.isnumeric():
            if int(letter) in tallListe:
                continue
            else:
                tallListe.append(int(letter))
    tallListe.sort()
    return tallListe
```

8 Oppgave 2.3 (5%)

Lag funksjonen ***read_game_file*** som har en input-parameter ***filename*** som er en tekststreng som spesifiserer navnet på fila det skal leses fra.

Funksjonen skal lese fra en tekstfil og konvertere hver linje til et innslag i en dictionary. Data på linjene i tekstfila er avskilt med semikolon (;). Det første ordet som står på linja skal være nøkkel, mens de resterende elementene på linja legges inn i en liste tilhørende nøkkelen. Hvis det allerede finnes et element med samme nøkkel, skal elementene legges til listen for den angitte nøkkelen.

Funksjonen skal returnere en dictionary som svarer til innholdet i tekstfila.

Eksempel på fil (gamefile.txt):

PS2;Gran Turismo 4;Final Fantasy X;Ico
GameCube;Metroid Prime;Resident Evil 4;Super Mario Sunshine
XBOX;Halo:Combat Evolved;Project Gotham Racing
PS2;SSX;Okami;Shadow of the Colossus
GameCube;Mario Kart:Double Dash!;Super Smash Bros.Melee
XBOX;Fable;Halo 2

Eksempel på kjøring av kode med fila gamefile.txt som ovenfor:

Shell

```
>>> d=read_game_file('gamefile.txt')
>>> print(d)
 {'PS2': ['Gran Turismo 4', 'Final Fantasy X', 'Ico', 'SSX', 'Okami', '
 Shadow of the Colossus'], 'GameCube': ['Metroid Prime', 'Resident Evil
 4', 'Super Mario Sunshine', 'Mario Kart:Double Dash!', 'Super Smash Br
 os.Melee'], 'XBOX': ['Halo:Combat Evolved', 'Project Gotham Racing', '
 Fable', 'Halo 2']}
>>> print(d['GameCube'])
 ['Metroid Prime', 'Resident Evil 4', 'Super Mario Sunshine', 'Mario Ka
 rt:Double Dash!', 'Super Smash Bros.Melee']
>>> print(d['PS2'])
 ['Gran Turismo 4', 'Final Fantasy X', 'Ico', 'SSX', 'Okami', 'Shadow o
 f the Colossus']
>>> print(d['XBOX'])
 ['Halo:Combat Evolved', 'Project Gotham Racing', 'Fable', 'Halo 2']
>>>
```

Skriv ditt svar her

```
1  def read_game_file(filename):
2      gameDict = {}
3      with open(filename, "r") as filestream:
4          for line in filestream:
5              lineList = line.split(";")
6              if lineList[0] not in gameDict:
7                  gameDict[lineList[0]] = []
8              gameDict[lineList[0]] +=(lineList[1:])
9      return gameDict
```

9 Oppgave 2.4 (10%)

Lag funksjonen ***write_diary*** som har en input-parameter **filename** som er en tekststreng som angir navn på en tekstfil.

Funksjonen skal gi bruker mulighet til å skrive en dagbok som skal lagres til en tekstfil med navnet **filename**. Funksjonen skal først skrive følgende til konsoll: *"Press enter without text on date to quit!"*. Så skal funksjonen spørre brukerene om å skrive inn dato (date), overskrift (title) og detaljer (details). Funksjonen skal la bruker skrive inn så mange innslag hun eller han vil, fram til at bruker trykker enter på spørsmål om dato uten å skrive noe tekst.

Funksjonen skal også sjekke om dato er skrevet inn på riktig format *dd.mm.yyyy*. Hvis bruker skriver inn dato på feil format, skal funksjonen skrive ut *"Incorrect date"* med innrykk (tab) til konsollet og be bruker om å skrive

inn dato på nytt.

For å sjekke om dato er skrevet på riktig format, skal hjelpefunksjonen ***correct_date*** lages som har en input-parameter ***date***, som er en tekststreng. Funksjonen skal returnere ***True*** hvis dato ser skrevet riktig på formatet dd.mm.yyyy, og ellers ***False***. Du skal sjekke om datoen er en gyldig dato, men du trenger ikke å sjekke for skuddår. Det betyr at bruker kan skrive dato til og med 29. for februar måned uansett år.

Eksempel på kjøring av kode der teksten i blått er skrevet inn av bruker:

Shell

```
>>> write_diary('my_diary.txt')
 Press enter without text on date to quit!

 Date [dd.mm.yyyy]: 2.desember 2020
         Incorrect date
 Date [dd.mm.yyyy]: 30.02.2020
         Incorrect date
 Date [dd.mm.yyyy]: 10.11.2020
 Subject: Mr C.
 Details: Who is Mr C. and why does he smell?

 Date [dd.mm.yyyy]: 11.11.2020
 Subject: Mr Ca.
 Details: Mr C. ate all my food. I'm not happy!!!

 Date [dd.mm.yyyy]: 12.11.2020
 Subject: Mr Cat
 Details: Mr C. is a cat... That is why he ate my house! Arrrg!

 Date [dd.mm.yyyy]:
 my_diary.txt saved to disk
>>>
```

Resulterende tekstfil med navn *filename* skal formateres på følgende måte (dato, tittel, skilletegn og detaljer) hvis teksten i blått overfor har blitt skrevet inn:

```
10.11.2020  Mr C.
==================
Who is Mr C. and why does he smell?

11.11.2020  Mr Ca.
====================
Mr C. ate all my food. I'm not happy!!!

12.11.2020  Mr Cat
====================
Mr C. is a cat... That is why he ate my house! Arrrg!
```

Det skal være like mange skilletegn (=) som totalt antall tegn i tittel-linja og det skal være et innrykk (tab) mellom dato og tittel.

Skriv ditt svar her

```python
def write_diary(filename):
    print("Press enter without text on date to quit!")

    while True:
        dateInput = input("Date [dd.mm.yyyy]: ")
        if (dateInput == ""):
            break
        if not correct_date(dateInput):
            print("     Incorrect date")
            continue


        titleInput = input("Title: ")
        detailsInput = input("Details: ")

        titleLine = dateInput + " " + titleInput
        spacerLine = ""
        for i in titleLine:
            spacerLine += "="

        with open(filename, "a") as filestream:
            filestream.write(titleLine + "\n")
            filestream.write(spacerLine + "\n")
            filestream.write(detailsInput+ "\n")
            filestream.write("\n")

def correct_date(date):
    dateList = date.split(".")
    if len(dateList) != 3: return False #Sjekker at det er riktig format med punktum
    for date in dateList:
        if not date.isdigit(): return False #Sjekker om dato inneholder bokstaver

    if int(dateList[1]) < 1 or int(dateList[1]) > 12: return False #Sjekker om måned er mellom 1-12
    if len(dateList[0]) != 2: return False #Sjekker om dato er 2 sifret
    if len(dateList[1]) != 2: return False #Sjekker om dato er 2 sifret
    if len(dateList[2]) != 4: return False #Sjekker om dato er 2 sifret

    mndLengder = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    dager = (mndLengder[int(dateList[1]) - 1])
    if int(dateList[0]) < 1 or int(dateList[0]) > dager: return False #Sjekker om riktig antall
        dager
    return True
```

10

Oppgave 3.1 (5%)

Vi representerer internt hvert kort som et 2-tuppel (v,f).
Her er v et tall fra 2 til 14 (14 er ess), og f er et av tegnene 's','h','d','c',
for henholdsvis spar (spades), hjerter (hearts), ruter (diamonds) og kløver (clubs).

a) Skriv funksjonen ***new_deck*** uten input-parametre, som returnerer en liste med alle 52 kort, representert som tupler beskrevet over. Rekkefølgen spiller ingen rolle.

b) Skriv funksjonen ***shuffle*** som tar en input-parameter liste, som er en liste (muterbar sekvens). Den skal returnere en liste med de samme elementene, men i en annen og tilfeldig rekkefølge.

Eksempel på kjøring av kode:

Shell

```
>>> deck = new_deck()
>>> deck
[(2, 's'), (3, 's'), (4, 's'), (5, 's'), (6, 's'), (7, 's'), (8, 's'), (9, 's'), (10, 's'), (11, 's'),
(12, 's'), (13, 's'), (14, 's'), (2, 'h'), (3, 'h'), (4, 'h'), (5, 'h'), (6, 'h'), (7, 'h'), (8, 'h'),
(9, 'h'), (10, 'h'), (11, 'h'), (12, 'h'), (13, 'h'), (14, 'h'), (2, 'd'), (3, 'd'), (4, 'd'), (5, 'd')
, (6, 'd'), (7, 'd'), (8, 'd'), (9, 'd'), (10, 'd'), (11, 'd'), (12, 'd'), (13, 'd'), (14, 'd'), (2, 'c
'), (3, 'c'), (4, 'c'), (5, 'c'), (6, 'c'), (7, 'c'), (8, 'c'), (9, 'c'), (10, 'c'), (11, 'c'), (12, 'c
'), (13, 'c'), (14, 'c')]
>>> shuffled_deck = shuffle(deck)
>>> shuffled_deck
[(3, 'd'), (6, 'c'), (8, 'd'), (6, 'h'), (6, 's'), (2, 'h'), (9, 'c'), (13, 'c'), (7, 'c'), (5, 'c'), (
9, 'd'), (7, 'd'), (14, 's'), (12, 'h'), (2, 's'), (10, 'd'), (4, 's'), (8, 'c'), (12, 's'), (11, 'h'),
(2, 'c'), (6, 'd'), (4, 'd'), (9, 'h'), (12, 'c'), (14, 'h'), (5, 'd'), (3, 'c'), (3, 's'), (11, 'c'),
(3, 'h'), (8, 's'), (13, 'h'), (7, 's'), (11, 's'), (9, 's'), (13, 's'), (10, 'h'), (8, 'h'), (10, 's')
, (4, 'c'), (11, 'd'), (4, 'h'), (7, 'h'), (10, 'c'), (14, 'c'), (5, 'h'), (2, 'd'), (13, 'd'), (12, 'd
'), (5, 's'), (14, 'd')]
>>>
```

Skriv ditt svar her

```python
import random

def new_deck():
    cardTypes = ["s", "h", "d", "c"]
    deck = []
    for cardType in cardTypes:
        for number in range(2, 15):
            deck.append((number, cardType))

    return deck

def shuffle(deck):
    buffer_deck = deck
    random.shuffle(buffer_deck)
    return buffer_deck

```

11

Oppgave 3.2 (10%)

Her skal vi skrive tre funksjoner som skal bestemme verdien til et spill med tre kort.
De tar alle en input-parameter "play", som er en liste med tre kort.

a) ***check_values*** skal returnere 5 hvis det er tre like kort, 4 hvis det er en straight, 2 hvis det er et par, og 1 hvis ingen av delene.
b) ***check_suits*** skal returnere 3 hvis alle har samme farge, 1 ellers.
c) ***evaluate_play*** skal returnere verdiene til play i henhold til verditabellen under.

Verdiene til et spill

Navn på spill	Verdi	Beskrivelse	Eksempel
Straight flush	6	En straight og en flush	3s, 4s, 5s
Tre like	5	Alle tre kort har samme verdi	8s, 8h, 8c
Straight	4	Tre påfølgende verdier	Th, Js, Qc
Flush	3	Alle i samme farge, men ikke straight	2h, 6h, Kh
Par	2	To kort lik verdi, men ikke tre like	Ac, Ah, Ks
Ingenting	1	Ingen av de over	Ac, Kh, Tc

Eksempel på kjøring:

Shell ×

```
>>> play = [(4,'s'), (5,'s'), (6, 's')]
>>> check_values(play)
4
>>> check_suits(play)
3
>>> evaluate_play(play)
6
>>>
```

Skriv ditt svar her

```
def check_values(play):
    if (play[0][0] == play[1][0] and play[1][0] == play[2][0]):
        return 5
    elif (play[0][0] == play[1][0] or play[1][0] == play[2][0] or play[0][0] == play[2][0]):
        return 2
    elif play[0][0]+1 == play[1][0] and play[1][0] +1 == play[2][0]:
        return 4
    else:
        return 1


def check_suits(play):
    if play[0][1] == play[1][1] and playclear[1][1] == play[2][1]:
        return 3
    else:
        return 1



def evaluate_play(play):
    values = check_values(play)
    suits = check_suits(play)

    if (values == 4 and suits == 3): return 6
    elif (values == 5): return 5
    elif (values == 4): return 4
    elif (suits == 3): return 3
    elif (values == 2): return 2
    else: return 1
```

12

Oppgave 3.3 (10%)

Skriv funksjonen ***computer_play,*** som tar en input-parameter **hand**, som er en liste med 5 kort. Den skal returnere det beste spillet (bestående av tre kort) som kan velges fra de fem kortene. Hvis det er flere spill som gir høyeste mulige verdi, så kan du velge fritt blant disse. Dette inkluderer tilfellet at ingen av spillene gir noe annet enn "ingenting".

Funksjonen skal også fjerne de tre spilte kortene fra input-argumentet hand, slik at denne kun inneholder to kort når funksjonen returnerer.

Greit å vite: Tre kort valgt fra fem kort kan gjøres på 10 forskjellige måter. Hvis vi representerer 5 kort med 12345, så har vi disse mulige spillene:
123, 124, 125, 134, 135, 145, 234, 235, 245, 345.

Eksempel på kjøring:

Shell

```
>>> hand1 = [(7,'c'), (4, 's'), (9, 'c'), (4, 'c'), (6, 'h')]
>>> computer_play(hand1)
[(7, 'c'), (9, 'c'), (4, 'c')]
>>> hand1
[(4, 's'), (6, 'h')]
>>>
```

Skriv ditt svar her

```
def computer_play(hand):
    playList = []

    playList.append((evaluate_play([hand[0],hand[1],hand[2]]),[hand[0],hand[1],hand[2]])) #123
    playList.append((evaluate_play([hand[0],hand[1],hand[3]]),[hand[0],hand[1],hand[3]])) #124
    playList.append((evaluate_play([hand[0],hand[1],hand[4]]),[hand[0],hand[1],hand[4]]))  #125
    playList.append((evaluate_play([hand[0],hand[2],hand[3]]),[hand[0],hand[2],hand[3]])) #134
    playList.append((evaluate_play([hand[0],hand[2],hand[4]]),[hand[0],hand[2],hand[4]])) #135
    playList.append((evaluate_play([hand[0],hand[3],hand[4]]),[hand[0],hand[3],hand[4]])) #145
    playList.append((evaluate_play([hand[1],hand[2],hand[3]]),[hand[1],hand[2],hand[3]])) #234
    playList.append((evaluate_play([hand[1],hand[2],hand[4]]),[hand[1],hand[2],hand[4]]))#235
    playList.append((evaluate_play([hand[1],hand[3],hand[4]]),[hand[1],hand[3],hand[4]])) #245
    playList.append((evaluate_play([hand[2],hand[3],hand[4]]),[hand[2],hand[3],hand[4]])) #345

    biggestPlay = (0,[])
    for play in playList:
        if play[0] > biggestPlay[0]:
            biggestPlay = play

    for card in biggestPlay[1]:
        cardIndex = hand.index(card)
        hand.pop(cardIndex)

    return biggestPlay[1]
```

13

Oppgave 3.4 (10%)

Skriv funksjonen ***human_play***, som tar en input-parameter **hand**, som er en liste med 5 kort.

Funksjonen skal vise spillerens kort på hånden, og be spilleren velge hvilke kort han vil spille.

Du kan selv velge nøyaktig hva du vil skrive som "prompt" til brukeren, men kortene på hånden skal printes på følgende måte:

Hvert kort skrives som streng med tegn, det første representerer verdien

med ett mellomrom mellom hvert kort. Eksempel:

Intern representasjon som 2-tuppel	skal vises slik
[(3,'c'),(4,'s'),(11,'h'),(12,'h'),(14,'d')]	3c 4s Jh Qh Ad

Input fra spilleren skal bestå av tre heltall fra 1 til 5, skilt med mellomrom. Tallene angir posisjonene til de tre kortene som skal spilles fra hånden. 1 betyr første kort som er listet, 2 det andre kortet osv. Brukeren spørres på nytt hvis valget ikke er gyldig, slik som spesifisert over.

Funksjonen returnerer en liste med de tre valgte kortene, hvert kort representert på internformatet (ikke indeksene).

Spilte kort skal fjernes fra input-parameter, slik at denne kun inneholder to kort når funksjonen returnerer.

Eksempel:

```
Shell
>>> hand1 = [(7,'c'), (4, 's'), (9, 'c'), (4, 'c'), (6, 'h')]
>>> human_play(hand1)
 Dine kort på hånden er 7c 4s 9c 4c 6h. Velg kort: 1 2
 Feil antall kort. Prøv på nytt!
 Dine kort på hånden er 7c 4s 9c 4c 6h. Velg kort: 1 1 2
 Feil antall kort. Prøv på nytt!
 Dine kort på hånden er 7c 4s 9c 4c 6h. Velg kort: 0 2 3
 Du må angi tre tall fra 1 til 5. Prøv på nytt!
 Dine kort på hånden er 7c 4s 9c 4c 6h. Velg kort: a 2 3
 Feil i innlesning. Prøv på nytt!
 Dine kort på hånden er 7c 4s 9c 4c 6h. Velg kort: 1 4 5
[(7, 'c'), (4, 'c'), (6, 'h')]
>>> hand1
[(4, 's'), (9, 'c')]
>>>
```

Skriv ditt svar her

```python
def human_play(hand):
    while True:
        flag = False

        displayStr = "Dine kort på hånden er "

        for card in hand:
            if card[0] <= 10:
                displayStr += str(card[0])
            elif card[0] == 11:
                displayStr += "J"
            elif card[0] == 12:
                displayStr += "Q"
            elif card[0] == 13:
                displayStr += "K"
            elif card[0] == 14:
                displayStr += "A"
            displayStr += card[1] + " "

        cardInput = input(displayStr + " Velg kort: ")
        cardNumbers = cardInput.split(" ")

        if len(cardNumbers) != 3:
            print("Feil antall kort. Prøv på nytt!")
            flag = True
            continue

        for number in cardNumbers:
            if number.isalpha():
                print("Feil i innlesning. Prøv på nytt!")
                flag = True
                break
            if int(number) < 1 or int(number) > 5:
                print("Du må angi tre tall fra 1 til 5. Prøv på nytt")
                flag = True
                break


        if not flag:
            break

    playCards = []
    for number in cardNumbers:
        playCards.append(hand[int(number) - 1])


    for card in playCards:
        cardIndex = hand.index(card)
        hand.pop(cardIndex)

    return playCards
```

14

Oppgave 3.5 (5%)

Skriv funksjonen ***update_winner*** som tar fire (4) input-parametere:
- ***won_cards***, som er en liste med kort som vunnet så langt i spillet
- ***winner_play***, en liste med kortene som ble spilt av en av spillerne.
- ***loser_play***, en liste med kortene som ble spilt av den andre spillern.
- ***war_cards***, som er en liste med kortene som ligger på bordet fra forrige rundet etter krig

Hvert kort er representert ved et 2-tuppel som beskrevet tidligere.

Funksjonen skal oppdatere input-parametrene ***won_cards*** og ***war_cards***.
Kortene i **winner_play**, **loser_play** og **war_cards** skal legges til **won_cards**.
war_cards skal tømmes for kort.
Input-parametrene **winner_play** og **loser_play** skal ikke endres.

Det kan være nyttig å huske på følgende: Hvis du skal fjerne aller elementene i en liste
som er sendt som input-parameter, så vil ikke liste = [] gjøre jobben.

Eksempel:
Anta at en av spillerne (f.eks. datamaskinen) vinner en runde med kortene i c_play (straight flush) mot kortene i h_play (human). (Det er straight flush mot flush). Anta videre at datamaskinen har vunnet 6 kort tidligere, og at det ligger 6 kort i war bunken.

c_won = [(14, 'c'), (14, 's'), (14, 'd'), (10, 's'), (4, 's'), (3, 'h')]
c_play = [(11, 'd'), (10, 'd'), (9, 'd')]
h_play = [(7,'c'), (9, 'c'), (4, 'c')]
war_cards = [(2, 'h'), (12, 'c'), (8, 'c'), (13, 'c'), (12, 'h'), (6, 'c')]

Resultat etter av et funksjonskall blir da:

Shell ×

```
>>> c_won = [(14, 'c'), (14, 's'), (14, 'd'), (10, 's'), (4, 's'), (3, 'h')]
>>> c_play = [(11, 'd'), (10, 'd'), (9, 'd')]
>>> h_play = [(7,'c'), (9, 'c'), (4, 'c')]
>>> war_cards = [(2, 'h'), (12, 'c'), (8, 'c'), (13, 'c'), (12, 'h'), (6, 'c')]
>>> update_winner(c_won, c_play, h_play, war_cards)
>>> c_won
[(14, 'c'), (14, 's'), (14, 'd'), (10, 's'), (4, 's'), (3, 'h'), (2, 'h'), (9, 'c'),
(7, 'c'), (8, 'c'), (12, 'h'), (13, 'c'), (9, 'd'), (6, 'c'), (10, 'd'), (11, 'd'),
(12, 'c'), (4, 'c')]
>>> war_cards
[]
>>>
```

Skriv ditt svar her

```
def update_winner(won_cards, winner_play, loser_play, war_cards):
    for card in winner_play:
        won_cards.append(card)
    for card in loser_play:
        won_cards.append(card)
    for card in war_cards:
        won_cards.append(card)
    war_cards.clear()

```

15

Oppgave 3.6 (10%)

Skriv funksjonen ***main***. Den tar ingen input-argumenter. Den skal gjennomføre et helt spill mellom brukeren (human player) og datamaskinen (computer player).

Main skal initialisere data: Lage en blandet kortstokk og sette start-data for hver spiller.

For hver runde skal

- det trekkes kort til spillerne
- bruker informeres om kortene på sin hånd og la ham/hum velge spill
- datamaskinen velge spill
- bestemme og informere om utfallet av runden
- oppdatere data.

Når alle rundene er spilt, skal funksjonen finne ut hvem som er vinner, eventuelt om det er uavgjort, og skrive ut informasjon om dette til brukeren.

Nøyaktig hva og hvordan du vil printe ut informasjon er ikke spesifisert, men må inneholde informasjon om hvilke kort som ble spilt for hver runde, hvem som vant runden, og om det ligger kort på bordet til neste runde etter "krig". Til slutt skal det printes ut informasjon resultatet, inklusive hvor mange kort hver spiller vant.

Eksempel på kjøring:

Kortene er blandet. La spillet begynne!

Runde 1. Dine kort på hånden er 6d 3h 9s Qd 9c. Velg kort: 2 3 5
Du spilte 3h 9s 9c. Datamaskinen spilte Ah 5s As.
Det er par mot par. Ingen vant runden.
Det ligger 6 på bordet.

Runde 2. Dine kort på hånden er 6d Qd 3d 8d 5c. Velg kort: 2 3 4
Du spilte Qd 3d 8d. Datamaskinen spilte 4c 2c Tc.
Det er flush mot flush. Ingen vant runden.
Det ligger 12 på bordet.

Runde 3. Dine kort på hånden er 6d 5c 7d 3c Jd. Velg kort: 1 2 3
Du spilte 6d 5c 7d. Datamaskinen spilte Ts Js 2s.
Det er straight mot flush. Du vant runden!

(runde 4 til 6 fjernet fra eksemplet)

Runde 7. Dine kort på hånden er Qh Td 6c Th 8s. Velg kort: 1 2 4
Du spilte Qh Td Th. Datamaskinen spilte 9h 5d 9d.
Det er par mot par. Ingen vant runden.
Det ligger 6 på bordet.

Runde 8. Siste runde!
Dine kort på hånden er 6c 8s 8c 2h Kd. Velg kort: 1 2 3
Du spilte 6c 8s 8c. Datamaskinen spilte Ac Kc Qc.
Det er par mot straight flush. Datamaskinen vant runden!

Du vant 36 kort. Datamaskinen vant 12 kort.

Du vant sammenlagt.

Skriv ditt svar her

```
def main():
    deck = shuffle(new_deck())

    h_hand = draw_cards(deck, 2)
    h_points = 0
    h_play = []

    c_hand = draw_cards(deck, 2)
    c_point = 0
    c_play = []

    while True:
        h_hand += (draw_cards(deck, 3))
        c_hand += (draw_cards(deck, 3))



        h_play = human_play(h_hand)
        c_play = computer_play(c_hand)

        print("Du spilte:" + str(h_play) + "Datamaskinen spilte: " + str(c_play))
```