NTNU
Kunnskap for en bedre verden

KANDIDAT

Le Christian (cl)

PRØVE

H2020 - Insperaøving 2, TDT4110 B

Emnekode	--
Vurderingsform	--
Starttid	26.10.2020 07:00
Sluttid	30.10.2020 16:00
Sensurfrist	--
PDF opprettet	28.10.2020 00:05

Forside

Oppgave	Tittel	Oppgavetype
i	Forside	Dokument

Teori

Oppgave	Tittel	Oppgavetype
1	Kryptering	Langsvar
2	Switching	Langsvar
3	Pipelining	Langsvar
4	Minnehieraki	Dra og slipp
5	Paritet	Langsvar

Akronym

Oppgave	Tittel	Oppgavetype
6	Akronym - del 1	Programmering
7	Akronym - del 2	Programmering

WhatUserLikes

Oppgave	Tittel	Oppgavetype
8	Det bruker liker - del 1	Programmering
9	Det bruker liker - del 2	Programmering
10	Det bruker liker - del 3	Programmering

Sangkonkurranse

Oppgave	Tittel	Oppgavetype
11	Sangkonkurranse - del 1	Programmering
12	Sangkonkurranse - del 2	Programmering

Memory game

Oppgave	Tittel	Oppgavetype
13	Memory game - del 1	Programmering
14	Memory game - del 2	Programmering
15	Memory game - del 3	Programmering

Derivasjon

Oppgave	Tittel	Oppgavetype
16	Derivasjon - del 1	Programmering
17	Derivasjon - del 2	Programmering

1 Kryptering

Forklar kort forskjellen på kryptering med privat nøkkel (Private key systems) og offentlig nøkkel (Public key systems).

Skriv ditt svar her

Privat nøkkel er en unik nøkkel, der kun et individet har sin private nøkkel. En offentlig nøkkel er en nøkkel som alle har tilgang til, og alle individer har en offentlig nøkkel som er tilpasset seg. Kombinering av begge typene kryptering gjør at kryptert informasjon kan åpnes av riktig individ, der individet kan åpne den offentlige krypteringa med senderes offentlige nøkkel og den private krypteringa med sin egen private nøkkel.

2 Switching

Gjør rede for forskjellene mellom linjesvitsjing (circuit switching), meldingssvitsjing (message switching) og pakkesvitsjing (packet switching). Drøft fordeler og ulemper med de ulike svitsje-metodene.

Skriv ditt svar her

Moderne linjeswitching: Et midlertidig linje opprettes mellom to parter, der kun partene har tilgang til linja og brukes til kommunikasjon. Etter endt bruk elimineres linjen.

Fordeler: Uavbrutt og rask kommunikasjon mellom partene
Ulemper: Mye ressurser ettersom det er flere enkelte linjer (dyrere)

Meldingswitching: En form for linjeswitching, der enkelte hop-linjer reserveres for meldinga. Kan beskrives som flere linjeswitchinger etter hverandre.

Fordeler: Flere meldinger kan sendes samtidig gjennom et nettverk, selv om flere parter er med i nettverket. (Nettverket "lagrer" meldingen underveis)
Ulemper: Kan ikke brukes for å sende kontinuerlig informasjon, som lyd og video. Evt. nettverket kan være dyrt pga. flere enheter for informasjon å hoppe mellom.

Pakkeswitching: Informasjon sendt fra flere parter sendes gjennom en felles linje. Deres informasjon deles opp i pakker og sender for tur gjennom linja.

Fordeler: Billig, ettersom flere nettverk deler på samme linje.
Ulemper: Som meldingsswitching kan ikke levere kontinuerlig informasjon, og kan forårsake "packet-loss" pga. mange packets i linja.

3 Pipelining

Forklar kort hvordan pipelining fungerer, og hvorfor det kan øke antallet instruksjoner en CPU utfører hvert sekund.

Skriv ditt svar her

Ved å sende informasjon rett etter hverandre kan en prosess på utføres samtidig som prosessen på steget bak også utføres. Altså for hver gang steg fullføres outputter CPUen info, vil det også inputtes info som skal prosesseres. Hvor raskt CPUen gjennomfører stegene avgjøres av den tregeste prosessen i CPUen.

4 Minnehieraki

Sorter de ulike lagringsalternativene etter hastighet, der den raskeste skal være øverst.
Dra og slipp alternativene.

Registre

Cache

RAM

Sekundærminne

5 Paritet

Denne oppgaven handler om paritetsbit, som ikke er gjennomgått i forelesning, men er likevel en del av pensum. (Hint: se kapittel 8 i teoriboken)

Alice skal sende en melding til Bob. Meldingen hun skal sende består av 16 bit, og ser slik ut:

1	0	1	1
0	1	0	1
1	1	0	1
0	0	1	0

For å øke sjansen for at melding skal komme korrekt frem, kan Alice bruke paritetssjekking.
Med Single Parity Checking (SPC) vil meldingen se slik ut, der P er paritetsbiten:

1	0	1	1	P
0	1	0	1	P
1	1	0	1	P
0	0	1	0	P

Med Row and Column (RAC) sjekk, vil meldingen se slik ut, der P er paritetsbit.

1	0	1	1	P
0	1	0	1	P
1	1	0	1	P
0	0	1	0	P
P	P	P	P	P

Når Bob mottar meldingen fra Alice, må vi sjekke om det har skjedd noe galt under transport av meldingen. Meldingen (uten paritetsbit) ser nå slik ut:

1	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0

Anta at paritetsbit ikke har blitt endret, og at det er brukt partalls-paritet (even-parity).

Vil Bob finne ut om det har skjedd noe galt? Vil det være en forskjell på å bruke Single Parity Checking (SPC) og Row and Column (RAC) sjekk i dette tilfellet? Begrunn svaret ditt.

(Hint: Regn ut paritetsbit med både SPC og RAC på den sendte og den mottatte meldingen).

Skriv ditt svar her

Bob vil ikke finne ut om noe er galt, ettersom ingen av metodene SPC og RAC kommer til å oppgave feilen.

1	0	1	1	1	X	1	0	1	1	1
0	1	0	1	0	X	0	1	0	1	0
1	1	0	1	1	X	0	1	1	1	1
0	0	1	0	1	X	1	0	0	0	1
0	0	0	1	1	X	0	0	0	1	1

De fire øverste rutene i kolonnen lengst til venstre inneholder bitene for SPC, og vi ser her at de er like. Det samme gjelder for bitene som er inkludert for RAC. Grunnen til dette er at antallet partall for alle kolonner og rader forblir det samme. Når biten i 1x3 bytter plass med 3x3, gjør også 1x4 med 3x4, vi ser her at langs radene endres ikke antallet partall, og langst kolonnene gjør det ikke det heller pga, en annen bit langs kolonnene gjorde opp for en endring.

6

Akronym - del 1

I denne oppgaven skal du lage en funksjon, *input_strings().* Denne funksjonen skal be brukeren angi 4 tekst-strenger og lagre dem separat i en liste. Deretter skal funksjonen returnere listen.

Eksempel på kall:

```
>>> input_strings()

  Skriv inn en streng: Norges
  Skriv inn en streng: tekniske
  Skriv inn en streng: naturvitenskapelige
  Skriv inn en streng: universitet
['Norges', 'tekniske', 'naturvitenskapelige', 'universitet']
```

Merk!: Funksjonen skal *returnere* listen, **ikke** *printe* den ut til skjerm. Grunnen til at listen blir skrevet til skjerm i eksemplet over, er at returverdien automatisk blir skrevet til skjerm i Thonny. Som dere kan se, blir returverdien skrevet i en annen font en input- og print-setninger. Generelt er det verdt å merke seg om oppgaver spør om at noe skal *returneres* eller *printes*, og det er forventet at dere kan forskjellen på eksamen. Dvs. presiseringer som denne vil dere ikke møte på eksamen, eller i resten av dette oppgavesettet.

Skriv ditt svar her

```
def input_strings():
    collection=[]
    for i in range(4):
        word=input("Skriv inn en streng: ")
        collection.append(word)
    return collection
```

7

Akronym - del 2

Vi ønsker å bruke listen fra forrige oppgave til å danne et akronym. Et akronym er en forkortelse bestående av den første bokstaven i hver del av betegnelsen vi vil forkorte.

Lag en funksjon *acronym()* som gjenbruker funksjonen *input_strings()* til å lage en liste med tekst-elementer. Deretter skal funksjonen printe ut akronymet bestående av strengene i store bokstaver.

Eksempel på kjøring av *acronym()*:

```
>>> acronym()

  Skriv inn en streng: Norges
  Skriv inn en streng: tekniske
  Skriv inn en streng: naturvitenskaplige
  Skriv inn en streng: universitet
  NTNU
```

Skriv ditt svar her

```
def acronym():
    acronym=""
    for i in range(4):
        word=input("Skriv inn en streng: ")
        letters=list(word)
        acronym+=letters[0]
    return acronym
```

8 Det bruker liker - del 1

Lag funksjonen *do_user_like(items)* som tar inn en liste med ord (*items*) og spør brukeren om å oppgi et tall på en skala f.o.m. 1 t.o.m. 10 som gjenspeiler hvor mye brukeren liker hvert element i listen. Om brukeren oppgir et tall utenfor intervallet [1, 10], skal programmet presisere at tallet skal være innenfor det gyldige intervallet, og etterspørre et nytt tall for elementet (se Eksempel 2). Når funksjonen har hentet inn tall for alle elementene i listen, skal funksjonen returnere en liste med tupler der det første elementet i tuppelen er ordet fra listen *items* og det andre elementet er tallverdien som hører til dette elementet (se siste linje i Eksempel 1 og 2).

Du kan anta at bruker alltid oppgir heltall.

Eksempel 1:

```
>>> do_user_like(['Dogs', 'Cats', 'Chocolate', 'Pancakes', 'Ice cream'])
On a scale from 1 to 10 where 10 is the highest, how much do you like:
Dogs? 10
Cats? 8
Chocolate? 10
Pancakes? 9
Ice cream? 7
[('Dogs', 10), ('Cats', 8), ('Chocolate', 10), ('Pancakes', 9), ('Ice cream', 7)]
```

Eksempel 2:

```
>>> do_user_like(['Dogs', 'Cats', 'Chocolate', 'Pancakes', 'Ice cream'])
On a scale from 1 to 10 where 10 is the highest, how much do you like:
Dogs? 10000
You have to give a value in the interval [1, 10]. Try again.
Dogs? -9
You have to give a value in the interval [1, 10]. Try again.
Dogs? 100
You have to give a value in the interval [1, 10]. Try again.
Dogs? 10
Cats? 9
Chocolate? 9
Pancakes? 9
Ice cream? 9
[('Dogs', 10), ('Cats', 9), ('Chocolate', 9), ('Pancakes', 9), ('Ice cream', 9)]
```

Skriv ditt svar her

```python
def do_user_like(items):
    print("On a scale from 1 to 10 where 10 is the highest, how much do you like:")
    L=[]
    for I in items:
        ans=int(input(I + "? "))
        while ans<1 or ans>10:
            ans=int(input("You have to give a value in the interval [1, 10]. Try again. "))
        O=[]
        O.append(I)
        O.append(ans)
        O=tuple(O)
        L.append(O)
    return L
```

9 Det bruker liker - del 2

Lag funksjonen *get_prioritized_list(lst)* som tar inn en liste med tupler (*lst*) der hver tuppel består av et ord som første element og et tall som andre. Funksjonen skal returnere en **ny** liste bestående av de samme tuplene i sortert rekkefølge. Tuplene sorteres primært etter tallverdien deres, der den høyeste tallverdien kommer først i listen. Om noen tupler har samme tallverdi, skal de tuplene med samme tallverdi sorteres i alfabetisk rekkefølge, der ord som kommer først i alfabetet kommer først (se Eksempel 1, 2 og 3. I eksemplene viser første linje funksjonen kalt med en liste som input, og andre linje viser den returnerte listen gitt input-listen).

Eksempel 1, alle tupler har forskjellige tall:

```
>>> get_prioritized_list([('Dogs', 8), ('Cats', 9), ('Chocolate', 10), ('Pancakes', 7), ('Ice cream', 4)])
[('Chocolate', 10), ('Cats', 9), ('Dogs', 8), ('Pancakes', 7), ('Ice cream', 4)]
```

Eksempel 2, alle tupler har samme tallverdi:

```
>>> get_prioritized_list([('Dogs', 8), ('Cats', 8), ('Chocolate', 8), ('Pancakes', 8), ('Ice cream', 8)])
[('Cats', 8), ('Chocolate', 8), ('Dogs', 8), ('Ice cream', 8), ('Pancakes', 8)]
```

Eksempel 3, noen tupler har samme tallverdi, men ikke alle:

```
>>> get_prioritized_list([('Dogs', 10), ('Cats', 10), ('Chocolate', 8), ('Pancakes', 10), ('Ice cream', 9)])
[('Cats', 10), ('Dogs', 10), ('Pancakes', 10), ('Ice cream', 9), ('Chocolate', 8)]
```

Skriv ditt svar her

```
def get_prioritized_list(lst):
    LoT=lst
    LoT.sort()
    Sorted=[]
    for N in range(10,0,-1):
        for T in LoT:
            if T[1]==N:
                Sorted.append(T)
    return Sorted
```

10 Det bruker liker - del 3

Lag funksjonen *what_user_likes_best(items, num)* som tar inn en liste med ord, *items*, og et heltall, *num*. Funksjonen skal printe "Invalid number given" og avbryte uten å gjøre noe mer om *num* enten er mindre enn 1 eller større enn lengden av listen *items*. Om *num* er innenfor intervallet f.o.m. 1 t.o.m. lengden av *items*, skal funksjonen først spørre brukeren om å oppgi et tall på en skala f.o.m. 1 t.o.m. 10 som gjenspeiler hvor mye brukeren liker hvert element i listen. Deretter skal funksjonen printe en toppliste bestående av *num* elementer.

Topplisten som printes skal være bygd opp slik: Først skrives "Your top *num* are:" til skjerm, evt. "Your number 1 is:" om *num* er lik 1. Deretter skal en nummerert liste skrives til skjerm. Denne nummererte listen skal være bygd opp slik at elementene fra listen *items* med høyest oppgitt tall fra bruker kommer først. Om noen elementer har samme tallverdi, skal det elementet som kommer først i alfabetet være først i listen. (Se siste 4 linjene i Eksempel 1 og siste 2 linjene i Eksempel 2 for eksempler på hvordan topplisten skal se ut.)

Du kan bruke funksjonene fra del 1 og 2, uansett om du fullførte dem eller ikke.

Eksempel 1:

```
>>> what_user_likes_best(['Dogs', 'Cats', 'Chocolate', 'Pancakes', 'Ice cream'], 3)
On a scale from 1 to 10 where 10 is the highest, how much do you like:
Dogs? 10
Cats? 10
Chocolate? 8
Pancakes? 5
Ice cream? 6
Your top 3 are:
1. Cats
2. Dogs
3. Chocolate
```

Eksempel 2:

```
>>> what_user_likes_best(['Dogs', 'Cats', 'Chocolate', 'Pancakes', 'Ice cream'], 1)
On a scale from 1 to 10 where 10 is the highest, how much do you like:
Dogs? 8
Cats? 8
Chocolate? 6
Pancakes? 10
Ice cream? 2
Your number 1 is:
1. Pancakes
```

Skriv ditt svar her

```
def what_user_likes_best(items,num):
    if num<1 or num>len(items):
        print("Invalid number given")
        return
    L=do_user_like(items)
    P=get_prioritized_list(L)
    if num==1:
        print("Your number 1 is: ")
        print("1.", P[0,0])
    elif num>1:
        for place in range(1,num+1):
            n=str(place)+"."
            print(n,P[place-1][0])

what_user_likes_best(["Dogs","Cats","Chocolate","Pancakes","Ice cream"],3)
```

11 Sangkonkurranse - del 1

Lag funksjonen *pop_random_song(songs)* som tar inn en liste (*songs*), fjerner et tilfeldig element fra listen, og returnerer elementet som ble fjernet.

Listen som tas inn består av tupler der første element er 1-2 linjer fra en sang og andre element er det neste ordet i sangen. Et eksempel på en slik liste er vist under:

```
songs = [("You hear the door slam. And realize there's nowhere left to", "run"),
         ("Oh, I wanna dance with somebody. I wanna feel the",  "heat"),
         ("There's a fire starting in my heart. Reaching a fever", "pitch"),
         ("Hey, I just met you and this is crazy. But here's my", "number"),
         ("'Cause baby, you're a firework. Come on, show 'em what you're", "worth")]
# Denne listen kan dere bruke for å teste koden deres.
```

Eksempel på kjøring:

```
>>> pop_random_song(songs)
("There's a fire starting in my heart. Reaching a fever", 'pitch')
>>> pop_random_song(songs)
('Oh, I wanna dance with somebody. I wanna feel the', 'heat')
>>> pop_random_song(songs)
("Hey, I just met you and this is crazy. But here's my", 'number')
```

Skriv ditt svar her

```
1  import random
2
3  def pop_random_song(songs):
4      ListIndex=random.randint(0,len(songs)-1)
5      Line=songs[ListIndex]
6      songs.pop(ListIndex)
7      return Line
```

12 Sangkonkurranse - del 2

Lag funksjonen *song_contest(songs)* som tar inn en liste (*songs*). Funksjonen skal:

- Velge en tilfeldig sang/tupel fra listen *songs*
- Skrive ut det første elementet i tuppelen og be brukeren om hva neste ord er
- Om brukeren gjetter feil skal programmet fortsette å spørre om og om igjen helt til brukeren gir rett svar
- Om brukeren gjetter riktig skal programmet gi tilbakemelding om dette.

Etter at brukeren har gjettet riktig og fått tilbakemelding om dette, skal funksjonen gjøre følgende:

- Om det **ikke** er flere sanger igjen i listen *songs*, skal programmet gratulere brukeren med å ha klart alle sangene og avslutte
- Om det er flere sanger igjen i listen *songs*, skal programmet spørre brukeren om hen vil fortsette. Om brukeren **ikke** ønsker dette skal programmet avslutte. Om brukeren ønsker enda en sang, skal bruker få det (dvs. programmet starter fra toppen igjen)

Du kan bruke funksjonen *pop_random_song(songs)* fra forrige oppgave uansett om du fullførte den eller ikke.

Eksempel 1:

```
>>> song_contest(songs)
The lyrics are:
Hey, I just met you and this is crazy. But here's my
What is the next word? creditcard
Wrong guess. Try again.
What is the next word? money
Wrong guess. Try again.
What is the next word? number
Correct!
Do you want to go again? (y/n) n
Welcome back later :D
```

Eksempel 2, alle rett:

```
>>> song_contest(songs)
The lyrics are:
Hey, I just met you and this is crazy. But here's my
What is the next word? number
Correct!
Do you want to go again? (y/n) y
The lyrics are:
There's a fire starting in my heart. Reaching a fever
What is the next word? pitch
Correct!
Do you want to go again? (y/n) y
The lyrics are:
'Cause baby, you're a firework. Come on, show 'em what you're
What is the next word? worth
Correct!
Do you want to go again? (y/n) y
The lyrics are:
You hear the door slam. And realize there's nowhere left to
What is the next word? run
Correct!
Do you want to go again? (y/n) y
The lyrics are:
Oh, I wanna dance with somebody. I wanna feel the
What is the next word? heat
Correct!
Congratulation, music lover! You have managed to get every song lyric we had correct!
```

Skriv ditt svar her

```
def song_contest(songs):
    Lans=""
    while Lans!="n":
        Line=pop_random_song(songs)
        FirstLine=Line[0]
        LastWord=Line[1]
        print("The lyrics are: ")
        print("Hey, I just met you and this is crazy. But here's my")
        print(LastWord)
        ans=""
        while ans!=LastWord:
            ans=input("What is the next word? ")
            if ans!=LastWord:
                print("Wrong guess. Try again.")
        print("Correct!")
        if songs!=[]:
            Lans=input("Do you want to go again? (y/n) ")
        elif songs==[]:
            print("Congratulations, music lover! You have managed to get every song lyrics we had
                correct!")
            break
    if Lans=="n":
        print("Welcome back later :D")
```

13 Memory game - del 1

Lag funksjonen *smallify_words(objects)* som tar inn en liste (*objects*) bestående av ord, og returnerer en **ny** liste med de samme ordene i små bokstaver.

Eksempel på kjøring:

```
>>> smallify_words(["dog", "CaT", "SCREEN", "mOuSE", "pan"])
['dog', 'cat', 'screen', 'mouse', 'pan']
```

Skriv ditt svar her

```
def smallify_words(objects):
    lowered=[]
    for word in objects:
        lword=word.lower()
        lowered.append(lword)
    return lowered
```

14 Memory game - del 2

Lag funksjonen *get_five_objects()* som ber bruker oppgi 5 ord separert med semikolon (;), og returnerer en liste med disse fem ordene i **små** bokstaver. Om bruker ikke oppgir fem ord, skal funksjonen informere brukeren om at feil antall ord er oppgitt, hvor mange ord brukeren oppga, og at bruker må oppgi 5 ord. Dvs. noe ala. "Du må oppgi 5 ord, ikke 2".

Du kan bruke funksjonen *smallify_words(objects)* fra forrige oppgave uansett om du fullførte den eller ikke.

Eksempel på kjøring:

```
>>> get_five_objects()
 Enter five objects separated by ';': dog;CaT;SCREEN
 You were supposed to enter FIVE objects, not 3. Try again.
 Enter five objects separated by ';': dog;CaT
 You were supposed to enter FIVE objects, not 2. Try again.
 Enter five objects separated by ';': dog;CaT;SCREEN;mOuSE;pan
['dog', 'cat', 'screen', 'mouse', 'pan']
```

Merk! Bruker kan oppgi ord som inneholder store bokstaver, men funksjonen vil endre disse ordene til å bare inneholde små bokstaver.

Skriv ditt svar her

```

def get_five_objects():
    object=[]
    while len(object.split(;))!=5:
    object=input("Enter five objects seperated by "";"": ")
        if len(object.split(;))!=5:
            print("You were supposed to enter FIVE objects, not", len(object.split(;)),". Try again
                .")
    filtered=smallify_words(object)
    return filtered
```

15

Memory game - del 3

Lag funksjonen *play_game()*. Denne funksjonen skal be bruker oppgi fem ord separert med semikolon (;), for deretter å starte en minnelek etter at brukeren har gjort det. Om brukeren ikke oppgir fem ord, skal programmet informere brukeren om at feil antall ord er oppgitt, for deretter å igjen spørre brukeren om å oppgi fem ord. Når bruker har oppgitt fem ord begynner minneleken. I minneleken blir brukeren bedt om å gjette på et av ordene brukeren oppga. Om brukeren **ikke** gjetter korrekt, skal brukeren bli informert om dette og bedt om å gjøre et nytt gjett. Om brukeren klarer å gjette korrekt, skal ordet fjernes fra oversikten over korrekte ord, og brukeren skal bli informert om at gjettet var korrekt. Videre, om bruker gjettet korrekt, skal programmet enten etterspørre et nytt gjett eller avslutte. Programmet etterspør et nytt gjett om det er flere ord igjen i oversikten over korrekte ord, og avslutter om det ikke er det. Programmet skal også avsluttes om brukeren skriver 'quit' i stedet for å gjette på et nytt ord.

Dette programmet skal behandle ord som like om ordene er de samme, uavhengig av bruken av små og store bokstaver. Dvs. at "cat", "CAT", og "CaT" vil bli sett på som samme ord.

Du kan bruke funksjonene *smallify_words(objects)* og *get_five_objects()* fra de forrige oppgavene uansett om du fullførte dem eller ikke.

Eksempel 1:

```
>>> play_game()
 Enter five objects separated by ';': dog;CaT;SCREEN;mOuSE;pan
 We shall now play a little memory game. Can you remember all the words you gave me?
 If it gets to hard for you and you want to give up, write 'quit'
 What is your guess? DOGGO
 Sorry, that was not one of the words
 What is your next guess? DOG
 Congratulations! You remembered dog
 What is your next guess? screEn
 Congratulations! You remembered screen
 What is your next guess? MouSE
 Congratulations! You remembered mouse
 What is your next guess? cat
 Congratulations! You remembered cat
 What is your next guess? pancakes
 Sorry, that was not one of the words
 What is your next guess? pancake
 Sorry, that was not one of the words
 What is your next guess? cake
 Sorry, that was not one of the words
 What is your next guess? pan
 Congratulations! You remembered pan
 You did it! You remembered all the objects
```

Eksempel 2:

```
>>> play_game()
 Enter five objects separated by ';': dog;CaT
 You were supposed to enter FIVE objects, not 2. Try again.
 Enter five objects separated by ';': dog;CaT;SCREEN;mOuSE;pan
 We shall now play a little memory game. Can you remember all the words you gave me?
 If it gets to hard for you and you want to give up, write 'quit'
 What is your guess? dOggo
 Sorry, that was not one of the words
 What is your next guess? dogs
 Sorry, that was not one of the words
 What is your next guess? quit
```

Skriv ditt svar her

```
1
```

16

Derivasjon - del 1

I denne oppgaven skal du lage funksjonen *derivative(x, function)* som tar inn en integer *x* og en funksjon *function* som parameterverdier. *derivative(x, function)* skal returnere en approksimasjon (tilnærming) av den deriverte i x.

En approksimasjon til den deriverte i x kan regnes ut vha. formelen: $f'(x) \approx \frac{f(x+h)-f(x)}{h}$, hvor h er et lite tall som utgjør endringen i x, Δx.
For å få en god approksimasjon bruk h = 1e-8 (=0.00000001).

Skriv ditt svar her

```
1
```

17

Derivasjon - del 2

Bruk *derivative(x, function)* fra forrige oppgave til å approksimere den deriverte til funksjonen $f(x) = x^2 + 2x + 13$, i punkt x = 3. Print ut approksimasjonen med to desimaler.

Skriv ditt svar her

```
1
```