Forside

Institutt for datateknologi og informatikk

Eksamensoppgave i TDT4110 - Informasjonsteknologi, grunnkurs

Faglige kontakter under eksamen:

- Børge Haugset (tlf.: 934 20 190)
- Yngve Dahl (tlf.: 905 27 892)

Eksamensdato: 1. desember 2018
Eksamenstid (fra-til): 09:00 – 13:00
Hjelpemiddelkode/Tillatte hjelpemidler: D

Annen informasjon:
Det er angitt i prosent hvor mye hver deloppgave i eksamenssettet teller ved sensur. Les gjennom hele oppgavesettet før du begynner å løse oppgaven. Disponer tiden godt!

Merk! Studenter finner sensur i Studentweb. Har du spørsmål om din sensur må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

1 Oppgave 1 - Teori (25%)

Marker det du mener er det mest riktige alternativet.

Feil svar gir IKKE minuspoeng.

1) Hvilket alternativ er IKKE et lag i TCP/IP-stabelen (stack)?

- Det fysiske laget (Physical layer)
- Sammenkoblingslaget (Connection layer)
- Internettlaget (Internet layer)
- Nettverksgrensesnitt-laget (Network layer)

2) Hvilken form for kanalkoding (channel coding) bruker Internettet?

- RAC (Read And Check)
- 16-bit sjekksum (16-bit checksum)
- SPC (Single Parity Checking)
- Hamming Coding

3) Hvilken metode brukes for å håndtere duplikater og at pakker kommer i feil rekkefølge?

- Sequencing (sekvensering)
- Flow control (flytkontroll)
- ACK (retransmisjon)
- Replay

4) Hva er "replay error" i nettverkssammenheng?

- At en bekreftelse (ACK) sendes dobbelt, noe som kan føre til at avsender tror en pakke har kommet fram selv om den ikke har det.
- At en forsinket etableringspakke oppretter en tidligere terminert forbindelse, og begynner å sende informasjonen på nytt.
- At en forsinket pakke fra tidligere sesjon blir akseptert i senere sesjon, og at korrekt pakke dermed blir avvist som duplikat.
- At det blir sendt duplikater til mottaker, som da må håndtere dette.

5) Hvordan håndteres en "replay error"

- Duplikater forkastes om en annen kopi av pakken alt er mottatt eller ligger i ventelisten
- Mottaker etterspør manglende pakker
- Pakkene merkes med et sekvensnummer
- Pakkene i en sesjon merkes med en unik ID

6) Hva er det som karakteriserer et DoS (Denial-of-Service) angrep?

- Angriperen gir brukeren feilaktig informasjon og lurer brukeren til å oppgi personlige opplysninger til angriperens gode.
- Angriperen oppgir feilaktige opplysninger om et produkt eller tjeneste eller leverer et produkt med lavere kvalitet enn forventet.
- Angriperen setter opp en firewall som brenner opp datamaskinen fra innsiden slik at maskinen ikke kan betjene legitime forespørsler.
- Angriperen sender enorme mengder pakker til en server slik at den ikke kan betjene legitime forespørsler.

7) Hvor mange bits består en IPv6-adresse av?

- 32
- 256
- 128
- 64

8) Hva spesifiserer “1”-ere i en subnett-maske?

- Hvor mange bit IP-adressen har.
- Hvilke bits av IP-adressen som utgjør suffiks.
- Nettverks-adressen som tilhører IP-adressen.
- Hvilke bits av IP-adressen som utgjør prefiks.

9) Hva skjer når en melding krypteres?

- Dataene i meldingen endres, slik at kun riktig mottaker kan rekonstruere den opprinnelige meldingen.
- Meldingen må gå gjennom en brannmur, slik at det blir vanskeligere for angripere å få tak i dataen.
- Dataene i meldingen deles i små pakker slik at mottaker er den eneste som får tak i dem alle.
- Meldingen blir sikret mot angrep fra "trojanske hester".

10) Hvordan fungerer kryptering med offentlig nøkkel (public key encryption)?

- En offentlig nøkkel brukes både til å kryptere og dekryptere en melding.
- Hver part får en hemmelig og en offentlig nøkkel. En melding kryptert med en offentlig nøkkel, kan kun dekrypteres med den korresponderende private nøkkelen.
- Partene deler en hemmelig nøkkel som brukes både for kryptering og dekryptering.
- En privat nøkkel brukes både til å kryptere og dekryptere en melding.

11) Hva var det som var så revolusjonerende med CPU?

- CPU gjorde det mulig å lagre data i maskinens minne som gir store fordeler med tanke på at programmer kan bli mer komplekse og endres fort kun ved å gi nye instruksjoner til minnet.
- CPU har gjort det mulig å benytte transistorer som er svært små og lette og i tillegg svært pålitelige. Dette gjør at maskinene blir mindre, billigere og mer pålitelige.
- CPU har gjort det mulig å implementere RAM.
- CPU har gjort det mulig for oss å digitalisere Punch Cards (Hullkort).

12) Hva sier Moores lov?

- Loven sier at klokkehastigheten øker proporsjonalt med antall programtellere.
- Loven sier at antall transistorer i en integrert krets dobles hvert 2. år
- Loven postulerer at: Ting vil gå galt uavhengig av situasjon hvis gitt muligheten.
- Loven sier at samplingsfrekvensen må være minst dobbelt så rask som den raskeste frekvensen.

13) Hvilke fem steg er med i “Fetch/Execute Cycle”?

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

14) Hvordan fungerer en transistor?

- Ligger som en bro mellom to plater som leder strøm, og bærer dermed med seg elektroner fra den ene platen til den andre.
- Enheten omformer 220V vekselstrøm til likestrøm som kan brukes til de ulike enhetene (CPU, lydkort, grafikkort, harddisk, RAM osv.) i datamaskinen
- En bryter som det enten kan gå strøm gjennom eller ikke, og som man kan styre ved hjelp av strøm.
- Fungerer ved at mange metalledninger (brushes) lager elektriske forbindelser med en metallkule.

15) CPU kalles også

- Integrated Circuit
- Microprocessor
- Processor
- Core

16) Hvor mange symboler kan representeres med 3 byte?

- 8
- 2048
- 4096
- 16777216

17) Hva er sant om "run-length-koding"?

- Run-length-koding er taps-komprimering, dvs. at den originale representasjonen ikke kan rekonstrueres eksakt fra den komprimerte versjonen.
- Run-length-koding er tapsløs komprimering, dvs. at den originale representasjonen av 0ere og 1ere kan bli rekonstruert perfekt fra den komprimerte versjonen.
- Run-length-koding er en form for komprimering hvor bit-sekvensene, både de tidligere og de kommende, avhenger av hverandre, og vil endres fortløpende etterhvert som mer data kodes.
- Run-length-koding vil alltid lønne seg, siden den reduserer antall bits som trengs for å lagre informasjonen til en tidel av det den var.

18) Extended ASCII (også kjent som ISO-8859-1) er

- 8 bits kode
- 7 bits kode
- 10 bits kode
- 9 bits kode

19) Hvilket binært tall representeres av det hexadesimale tallet 39A?

- 001110011011
- 001110011010
- 011111101
- 010110111100

20) 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
- 0110 1111 0100 0010 0110 1111
- 0100 0010 0110 0010 0100 0010
- 0110 0010 0100 0010 0110 0010

Maks poeng: 20

Useful Python functions and commands

Built-in:

format(numeric_value, format_specifier)

Formats a numeric value into a string according to the format specifier, which is a string that contains special characters specifying how the numeric value should be formatted. Examples of various formatting characters are “f=floating-point, e=scientific notation, %=percentage, d=integer”. A number before the formatting character will specify the field width. A number after the character “.” will format the number of decimals.

%

Remainder (modulo operator): Divides one number by another and gives the remainder.

len(s)

Return the length (the number of items) of a string, tuple, list, dictionary or other data structure.

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

Return a string containing a nicely printable representation of an object.

range(stop)

Returns a list with integers from 0, up to but not including stop. range(3) = [0, 1, 2]. Often used in combination with for loops: for i in range(10)

range([start], stop[, step])

start: Starting number of the sequence.

stop: Generate numbers up to, but not including, this number.

step: Difference between each number in the sequence.

chr(i)

Return a string of one character whose ASCII code is the integer i. For example, chr(97) returns the string 'a'. This is the inverse of ord()

ord()

Given a string of length one, return an integer representing the Unicode code point of the character when the argument is a unicode object, or the value of the byte when the argument is an 8-bit string. For example, ord('a') returns the integer 97.

tuple(iterable)

Accepts something iterable (list, range etc.) and returns the corresponding tuple.

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

Enters a loop with x being one and one item from iterable. idx is an integer containing the loop number.

try: except: else: finally:

```
try:
   # Code to test
except:
   # If code fails. Many exception types, like IOError for file operations.
   # Variant: except Exception as exc # Let's you print the exception.
else:
```

```
    # Runs if no exception occurs
finally:
    # Runs regardless of prior code having succeeded or failed.
```

String operations:

s.isalnum()

Returns true if the string contains only alphabetic letters or digits and is at least one character of length. Returns false otherwise.

s.isalpha()

Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.

s.isdigit()

Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.

s.isspace()

Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (\n), and tabs (\t)).

s.ljust(width)

Return the string left justified in a string of length width.

s.rjust(width)

Return the string right justified in a string of length width.

s.join(list)

Returns a string listing all items in the list, separated by s.

s.lower()

Returns a copy of the string with all alphabetic letters converted to lowercase.

s.upper()

Returns a copy of the string with all alphabetic letters converted to uppercase.

s.strip()

Returns a copy of the string with all leading and trailing white space characters (spaces, newlines and tabs) removed.

s.strip(char)

Returns a copy of the string with all instances of char that appear at the beginning and the end of the string removed.

s.split(str)

Returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified).

str.splitlines([keepends])

Return a list of the lines in the string, breaking at line boundaries. This method uses the universal newlines approach to splitting lines. Line breaks are not included in the resulting list unless keepends is given and true.

Python recognizes "\r", "\n", and "\r\n" as line boundaries for 8-bit strings.

s.endswith(substring)

The substring argument is a string. The method returns true if the string ends with substring.

s.startswith(substring)

The substring argument is a string. The method returns true if the string starts with substring.

s.find(substring)

The substring argument is a string. The method returns the lowest index in the string where substring is found. If substring is not found the method returns -1.

s.replace(old, new)

The old and new arguments are both strings. The method returns a copy of the string with all instances of old replaced by new.

*str.format(*args, **kwargs)*

Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces {}. Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

List operations:

s[i:j:k]

Return slice starting at position i extending to position j every k items. Can also be used for strings.

item in s

Determine whether a specified item is contained in a list.

min(list)

Returns the item that has the lowest value in the sequence.

max(list)

Returns the item that has the highest value in the sequence.

s.append(x)

Append new element x to end of s.

s.insert(index,item)

Insert an item into a list at a specified position given by an index.

s.index(item)

Return the index of the first element in the list containing the specified item.

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

Removes the first element containing the item.

s.reverse()

Reverses the order of the items in a list.

s.sort()

Rearranges the elements of a list so they appear in ascending order.

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.

d.items()

Returns all the keys in a dictionary and their associated values as a sequence of tuples.

d.keys()

Returns all the keys in a dictionary as a sequence of tuples.

d.pop(key, default)

Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

d.popitem()

Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

d.values()

Returns all the values in dictionary as a sequence of tuples.

File operations:

open()

Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing). Adding 'b' to the read or write attribute lets you use binary mode to save the value of variables to file and load from them. For binary to work you need to import the pickle library.

f.read(size)

Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.

f.readline()

Reads a single line from the file (reads until and including a newline character (\n) is found), and returns it as a string.

f.readlines()

Reads data from the file and returns it as a list of strings.

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes a sequence of strings (typically a list of strings) to file.

f.close()

Close the file and free up any system resources taken up by the open file.

Libary operations:

pickle.dump(obj, file)

Write a pickled representation of obj to the open file object file.

pickle.load(file_object)

Read a string from the open file object file and interpret it as a pickle data stream, reconstructing and returning the original object hierarchy.

Erstatt denne teksten med ditt innhold...

2 Oppgave 2a - Kodeforståelse (5%)

Funksjonen **myst** har følgende kode:

```
def myst(val1, val2):
    if (val1 and val2):
        return 1
    elif (val1 and not val2):
```

```
    return 2
elif (not val1 and val2):
    return 3
else:
    return 4
```

Hva returneres ved funksjonskallet under?
myst(((True and False) or (False and True)), ((False or True) and (not(not True))))

Velg ett alternativ

- 2
- 1
- 3
- 4

Maks poeng: 1

3 Oppgave 2b - Kodeforståelse (5%)

Funksjonen **sqr** skal ta inn en liste (**numbers**) bestående av heltall som parameter. Funksjonen skal endre annethvert heltall i listen (fra og og med andre element) til kvadratet av heltallet (heltallet multiplisert med seg selv). Til slutt skal funksjonen returnere den endrede listen.

Gitt funksjonen:

```
def sqr(numbers):
    #KODE 1
        numbers[x]=numbers[x]**2
    return numbers
```

Eksempel på kall av funksjonen **sqr**:

```
>>> numbers = sqr([2,4,6,8,10,12])
>>> print(numbers)
[2, 16, 6, 64, 10, 144]
```

Hvordan skal linjen med innholdet **#KODE1** i koden til **sqr** se ut for at funksjonen skal fungere på måten beskrevet over ved kjøring?
Velg ett alternativ

- for x in range(2,len(numbers)-1,2):
- for x in range(1,len(numbers)-1,2):
- for x in range(1,len(numbers),2):
- for x in range(len(numbers),2):
- for x in numbers:
- while x in range(1,len(numbers),2):

Maks poeng: 1

4 Oppgave 2c - Kodeforståelse (10%)

Funksjonen **prime_numbers** er ment å returnere en liste som inneholder alle primtall, dvs. tall som bare kan deles med seg selv og 1, i et bestemt intervall angitt av parametrene **start** og **stopp**. Anta at *start* alltid vil være større enn 0 og mindre enn *stopp*.

Gitt funksjonen:

```
def prime_numbers(start, stop):
    primes = []
    for num in range(start,stop + 1):
        if num > 1:
            prime = True
            for i in range(2,num):
                #KODE1
                    prime = False
                    break
            if prime:
                #KODE2
    return primes
```

Eksempel på kall av funksjonen **prime_numbers:**

```
>>> primes = prime_numbers(1, 16)
>>> print(primes)
[2, 3, 5, 7, 11, 13]
```

Hvordan skal linjene med innholdet **#KODE1** og **#KODE2** i koden til **prime_numbers** se ut for at funksjonen skal ha den tiltenkte virkemåten under kjøring?

1) Velg ett alternativ for #KODE1

- if (num//i) == 0:
- if num == i:
- if (num % i) == 0:
- if (num/i) == 0:
- if (num/i) == 1:
- for j in range(2,num):

2) Velg ett alternativ for #KODE2

- break
- primes + num
- primes.append(i)
- continue
- primes.append(num)
- primes.extend(num)

Maks poeng: 2

5 Oppgave 2d - Kodeforståelse (5%)

Funksjonen **palindrome** tar inn en streng som parameter og har til hensikt å sjekke om ordet eller uttrykket representert i strengen gir samme resultat enten det leses fra høyre eller venstre. Hvis ordet eller utrykket gir samme resultat skal funksjonen returnere *True*. Hvis ikke skal den returnere *False*.

Funksjonen skal kun returnere *True* i tilfeller hvor tegnsetting i strengen blir helt lik uavhengig av hvilken vei strengen leses. Det betyr at funksjonskallet **palindrome("Radar")** vil returnere *False* siden

funksjonen skiller på store og små bokstaver.

Gitt funksjonen:

```
def palindrome(s):
   #KODE1
```

Eksempel på kjøring av funksjonen **palindrome**:

```
>>> palindrome("radar")
True
```

Hvordan skal linjen med innholdet **#KODE1** (return-setningen) i koden til **palindrome** se ut for at funksjonen skal fungere som beskrevet over ved kjøring?

Velg ett alternativ

- return not(bool(s.find(s[0:len(s):])))
- return bool(s.find(s[len(s):1]))
- return not(bool(s.find(s[len(s):1])))
- return bool(s.find(s[len(s):1]))
- return bool(s.find(s[::-1]))
- return not(bool(s.find(s[::-1])))

Maks poeng: 1

6 Oppgave 2e - Kodeforståelse (5%)

Gitt funksjonen:

```
def myst(tall1,tall2,nr):
   for i in range(2,nr):
      nytt = tall1 + tall2
      tall1 = tall2
      tall2 = nytt
   return nytt
```

Hva skrives ut når følgende kode kjøres?

print(myst(0,1,7))

Velg ett alternativ

- 5
- 8.0
- 11
- 13
- 13.0
- 8

Maks poeng: 1

7 Oppgave 2f - Kodeforståelse (5%)

Oppgave 2: Rekursjonsstudie (6%)

Gitt funksjonen:

```
def myst(x,y):
    if x%y == 0:
        return y
    else:
        return myst(y,x%y)
```

Hva skrives ut når følgende kode kjøres?

print(myst(24,88))

Velg ett alternativ

- 8
- 6
- 16
- 4
- 12
- 18

Maks poeng: 1

Useful Python functions and commands

Built-in:

format(numeric_value, format_specifier)

Formats a numeric value into a string according to the format specifier, which is a string that contains special characters specifying how the numeric value should be formatted. Examples of various formatting characters are “f=floating-point, e=scientific notation, %=percentage, d=integer”. A number before the formatting character will specify the field width. A number after the character “.” will format the number of decimals.

%

Remainder (modulo operator): Divides one number by another and gives the remainder.

len(s)

Return the length (the number of items) of a string, tuple, list, dictionary or other data structure.

int(x)

Convert a string or number to a plain integer.

float(x)

Convert a string or a number to floating point number.

str([object])

Return a string containing a nicely printable representation of an object.

range(stop)

Returns a list with integers from 0, up to but not including stop. range(3) = [0, 1, 2]. Often used in combination with for loops: for i in range(10)

range([start], stop[, step])

start: Starting number of the sequence.

stop: Generate numbers up to, but not including, this number.

step: Difference between each number in the sequence.

chr(i)

Return a string of one character whose ASCII code is the integer i. For example, chr(97) returns the string 'a'. This is the inverse of ord()

ord()

Given a string of length one, return an integer representing the Unicode code point of the character when the argument is a unicode object, or the value of the byte when the argument is an 8-bit string. For example, ord('a') returns the integer 97.

tuple(iterable)

Accepts something iterable (list, range etc.) and returns the corresponding tuple.

if x in iterable:

Returns True if x is an item in iterable.

for idx, x in enumerate(iterable)

Enters a loop with x being one and one item from iterable. idx is an integer containing the loop number.

try: except: else: finally:

```
try:
   # Code to test
except:
   # If code fails. Many exception types, like IOError for file operations.
   # Variant: except Exception as exc # Let's you print the exception.
else:
```

```
    # Runs if no exception occurs
finally:
    # Runs regardless of prior code having succeeded or failed.
```

String operations:

s.isalnum()

Returns true if the string contains only alphabetic letters or digits and is at least one character of length. Returns false otherwise.

s.isalpha()

Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.

s.isdigit()

Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.

s.isspace()

Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (\n), and tabs (\t)).

s.ljust(width)

Return the string left justified in a string of length width.

s.rjust(width)

Return the string right justified in a string of length width.

s.join(list)

Returns a string listing all items in the list, separated by s.

s.lower()

Returns a copy of the string with all alphabetic letters converted to lowercase.

s.upper()

Returns a copy of the string with all alphabetic letters converted to uppercase.

s.strip()

Returns a copy of the string with all leading and trailing white space characters (spaces, newlines and tabs) removed.

s.strip(char)

Returns a copy of the string with all instances of char that appear at the beginning and the end of the string removed.

s.split(str)

Returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified).

str.splitlines([keepends])

Return a list of the lines in the string, breaking at line boundaries. This method uses the universal newlines approach to splitting lines. Line breaks are not included in the resulting list unless keepends is given and true.

Python recognizes "\r", "\n", and "\r\n" as line boundaries for 8-bit strings.

s.endswith(substring)

The substring argument is a string. The method returns true if the string ends with substring.

s.startswith(substring)

The substring argument is a string. The method returns true if the string starts with substring.

s.find(substring)

The substring argument is a string. The method returns the lowest index in the string where substring is found. If substring is not found the method returns -1.

s.replace(old, new)

The old and new arguments are both strings. The method returns a copy of the string with all instances of old replaced by new.

*str.format(*args, **kwargs)*

Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces {}. Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

List operations:

s[i:j:k]

Return slice starting at position i extending to position j every k items. Can also be used for strings.

item in s

Determine whether a specified item is contained in a list.

min(list)

Returns the item that has the lowest value in the sequence.

max(list)

Returns the item that has the highest value in the sequence.

s.append(x)

Append new element x to end of s.

s.insert(index,item)

Insert an item into a list at a specified position given by an index.

s.index(item)

Return the index of the first element in the list containing the specified item.

s.pop()

Return last element and remove it from the list.

s.pop(i)

Return element i and remove it from the list.

s.remove(item)

Removes the first element containing the item.

s.reverse()

Reverses the order of the items in a list.

s.sort()

Rearranges the elements of a list so they appear in ascending order.

Dictionary operations:

d.clear()

Clears the contents of a dictionary

d.get(key, default)

Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.

d.items()

Returns all the keys in a dictionary and their associated values as a sequence of tuples.

d.keys()

Returns all the keys in a dictionary as a sequence of tuples.

d.pop(key, default)

Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

d.popitem()

Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

d.values()

Returns all the values in dictionary as a sequence of tuples.

File operations:

open()

Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing). Adding 'b' to the read or write attribute lets you use binary mode to save the value of variables to file and load from them. For binary to work you need to import the pickle library.

f.read(size)

Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.

f.readline()

Reads a single line from the file (reads until and including a newline character (\n) is found), and returns it as a string.

f.readlines()

Reads data from the file and returns it as a list of strings.

f.write(string)

Writes the contents of string to file.

f.writelines(list)

Writes a sequence of strings (typically a list of strings) to file.

f.close()

Close the file and free up any system resources taken up by the open file.

Libary operations:

pickle.dump(obj, file)

Write a pickled representation of obj to the open file object file.

pickle.load(file_object)

Read a string from the open file object file and interpret it as a pickle data stream, reconstructing and returning the original object hierarchy.

The text file flerkamp.txt and multigames.txt has the following layout:

```
Name,   Poker, Highjump, Balloonshooting, SausageEating, HoldBreath
John,   8,     1.67,     17,              23,           2:01.65
Lisa,   12,    1.30,     12,              13,           1:13.02
Per,    8,     1.55,     8,               0,            1:51.35
Nelly,  2,     1.34,     9,               17,           0:31.18
Nora,   5,     1.87,     13,              5,            2:01.65
```

Task 3 - multigames: task description

Flerkamp - oppgavebeskrivelse

Du er med i arrangementskomitéen for de årlige flerkamplekene som arrangeres i bygården du bor i. Alle deltakere er med på fem ulike øvelser, og resultatene lagres i en fil på følgende format (se også nederst i funksjonshjelpen til venstre):

```
Name,    Poker,  Highjump,  Balloonshooting, SausageEating,  HoldBreath
John,    8,      1.67,      17,               23,            2:01.65
Lisa,    12,     1.30,      12,               13,            1:13.02
Per,     8,      1.55,      8,                0,             1:51.35
Nelly,   2,      1.34,      9,                17,            0:31.18
Jørn,    5,      1.87,      13,               5,             2:01.65
```

... listen fortsetter med én linje for hver deltaker. Du kan forutsette at alle deltakerne har verdier for alle øvelsene, selv om noen av dem kan ha 0. Du kan også forutsette at i senere oppgaver betyr *høyere verdi* i en øvelse et *BEDRE* resultat.

For å gi lesbarhet har denne filen litt spesiell struktur. Hvert element er separert med et komma ‘,’ og et **variabelt** antall mellomrom (whitespace) ‘ ‘.

Deltakernes navn er skrevet i første kolonne. **Deltakernes rekkefølge er ukjent.** Tilsvarende inneholder den første raden navnet på øvelsene. **Øvelsenes rekkefølge kan også variere (men den er lik for alle deltakerne)**. I de følgende oppgavene må dere derfor velge en løsning som tar hensyn til at man ikke kan vite at resultatene fra ballongskytingen kommer i kolonne 3 og liknende. Du kan derimot forutsette at resultatet til hver deltaker i en øvelse er i samme kollonne som øvelsens navn.

Eksisterende funksjon du kan bruke ved behov

Det eksisterer en allerede kodet funksjon dere kan bruke ved behov, kalt *sort_list*(mylist,elemnum). Den tar to parametre. Den første er en todimensjonal liste. Den andre parameteren bestemmer hvilken kolonne listen skal sorteres på. Funksjonen returnerer den samme listen, men med elementene sortert etter valgt elementnummer.

Eksempel på bruk av *sort_list(mylist,elemnum)*: Legg merke til at den sorterer på verdien av den andre elementet i hver subliste ved å kalle sort_list med parameteren 1.

```
>>> sort_list([['John', 8], ['Lisa', 12], ['Per', 8]],1)
[['Lisa', 12], ['John', 8], ['Per', 8]]
# Her kom ['Lisa', 12] først siden 12 var det høyeste tallet i kolonne 1
```

8 Oppgave 3a – Programmering (5%)

Skriv funksjonen **read_file** som tar en inn-parameter **filename**. Denne funksjonen skal lese inn en tekstfil **filename**, som bekriver resultatene fra årets flerkamp-konkurranse. Resultatene er lagret i tekstfilen *flerkamp.txt*, som ligger i den samme mappen (directory) som du skal lagre python-koden. Resultatene i tekstfilen er på formatet beskrevet over. Funksjonen skal returnere innholdet i tekstfilen i form av én lang streng. Dersom filen ikke finnes skal funksjonen skrive ut feilmeldingen "Kan ikke finne filen flerkamp.txt" til skjermen, og returnere verdien *None*.

Eksempel på kjøring av funksjonen og utskrift av returverdi:

```
>>> resultater = read_file('flerkamp.txt')
>>> resultater
'Name,    Poker,  Highjump,  Balloonshooting, SausageEating,  HoldBreath\nJohn,    8,
1.67,     17,               23,               2:01.65\nLisa,    12,     1.30,     12,               13,
          1:13.02\nPer,     8,      1.55,     8,                0,              1:51.35\nNelly,   2,
1.34,     9,                17,               0:31.18\nNora,    5,      1.87,     13,               5,
2:01.65\n'
```

Skriv ditt svar her...

```
1
```

Maks poeng: 10

9 Oppgave 3b - Programmering (5%)

Skriv funksjonen **list_from_string** som tar inn strengen **txt** som inn-parameter. Forutsett at **txt** har et format tilsvarende én enkelt linje i *flerkamp.txt*, dvs. et sett med verdier (navn og resultat per øvelse) separert med komma og et vilkårlig antall mellomrom (whitespace). Funksjonen skal returnere en liste med strenger som beskriver de enkelte verdiene. Eventuelle mellomrom, linjeskift eller tabulatorer (whitespace) i strengen må fjernes fra hvert element i listen før listen returneres.

Eksempel på bruk:

```
>>> list_from_string("Lisa,  12,   1.30,   12,    13,  1:13.02\n")
['Lisa', '12', '1.30', '12', '13', '1:13.02']
```

Skriv ditt svar her...

```
1
```

Maks poeng: 10

10 Oppgave 3c – Programmering (5%)

Skriv funksjonen **make_result_list** som tar inn strengen returnert av funksjonen **read_file** (Oppgave 3a) som inn-parameter. Funksjonen **make_result_list** skal returnere en to-dimensjonal liste der hvert listeelement er en liste som innholder verdiene fra hver linje i filen *flerkamp.txt.* Bruk gjerne funksjonen **list_from_string** (Oppgave 3b) i løsningen din.

Eksempel på bruk:

```
>>> string = read_file('flerkamp.txt')

>>> string
'Name,  Poker,  Highjump,  Balloonshooting, SausageEating,  HoldBreath\nJohn,  8,
1.67,  17,  23,  2:01.65\nLisa,  12,  1.30,  12,  13,
1:13.02\nPer,  8,  1.55,  8,  0,  1:51.35\nNelly,  2,  1.34,  9,
17,  0:31.18\nNora,  5,  1.87,  13,  5,  2:01.65'

>>> results = make_result_list(string)

>>> results

[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', '8',
'1.67', '17', '23', '2:01.65'], ['Lisa', '12', '1.30', '12', '13', '1:13.02'], ['Per', '8', '1.55', '8', '0',
'1:51.35'], ['Nelly', '2', '1.34', '9', '17', '0:31.18'], ['Nora', '5', '1.87', '13', '5', '2:01.65']]
```

Skriv ditt svar her...

```
1
```

Maks poeng: 10

11 Oppgave 3d – Programmering (6%)

Skriv funksjonen **time_to_seconds** som tar strengen **time** som inn-parameter. Strengen angir en deltakers sluttid i en øvelse (f.eks. holde pusten lengst mulig) og vil ha følgende format: *min:sek.hundredeler*. Funksjonen skal gjøre om strengen til et flyttall med formatet *sekunder.hundredeler* og returnere dette flyttallet.

Eksempel på bruk:

```
>>> time_to_seconds('2:01.65')

121.65

>>> print(type(time_to_seconds('2:21.65')))
```

<class 'float'>

Skriv ditt svar her...

```
1
```

Maks poeng: 10

12 Oppgave 3e – Programmering (9%)

Skriv funksjonen **str_to_numbers** som tar inn **results**, en to-dimensjonal liste, som parameter. Funksjonen skal kunne ta inn den to-dimensjonale listen som returneres fra funksjonen **make_result_list** (Oppgave 3c) og gjøre om tall på strengformat til heltall eller flyttall. Hvis et element har formatet '2:23.56' skal disse gjøres om til antall sekunder og antall hundredeler (altså her 143.56 som flyttall). Hvis elementet har formatet '1.3' skal det gjøres om til et flyttall, mens hvis strengen bare inneholder et tall som '12' skal den gjøres om til et heltall. Funksjonen **str_to_numbers** skal returnere den formaterte (to-dimensjonale) listen.

Eksempel på bruk:

>>> # results inneholder resultatet fra oppgave 3c, og den har følgende format:

>>> results # Dette er formatet fra oppgave 3c
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', '8', '1.67', '17', '23', '2:01.65'], ['Lisa', '12', '1.30', '12', '13', '1:13.02'], ['Per', '8', '1.55', '8', '0', '1:51.35'], ['Nelly', '2', '1.34', '9', '17', '0:31.18'], ['Nora', '5', '1.87', '13', '5', '2:01.65']]

>>> results = str_to_number(results)

>>> results # Dette er results fra oppgave 3e
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'],
['John', 8, 1.67, 17, 23, 121.65], ['Lisa', 12, 1.3, 12, 13, 73.02],
['Per', 8, 1.55, 8, 0, 111.35], ['Nelly', 2, 1.34, 9, 17, 31.18],
['Nora', 5, 1.87, 13, 5, 121.65]]

Skriv ditt svar her...

```
1
```

Maks poeng: 10

13 Oppgave 3f – Programmering (5%)

Skriv funksjonen **find_data**, som har input-parametrene **event**, **name** og **results**. Parametrene inneholder, i den rekkefølgen, en øvelse, deltagerens navn og resultatlisten for en fullstendig flerkamps-konkurranse. Denne listen er formattert slik som den returneres fra funksjonen **str_to_numbers** (i oppgave 3e). Navnet på øvelsene og deltagerne er i ukjent rekkefølge.

Funksjonen skal bruke listen til å finne resultatet for en deltager i den oppgitte øvelsen, og returnere denne verdien.

I eksempelet under er det vist hvordan resultatet (returverdien fra **str_to_numbers**) ser ut før **find_data** blir kjørt:

```
>>> results
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', 8, 1.67, 17, 23, 121.65], ['Lisa', 12, 1.3, 12, 13, 73.02], ['Per', 8, 1.55, 8, 0, 111.35], ['Nelly', 2, 1.34, 9, 17, 31.18], ['Nora', 5, 1.87, 13, 5, 121.65]]

>>> find_data('SausageEating','John',results)
```

23

Skriv ditt svar her...

```
1
```

Maks poeng: 10

14 Oppgave 3g - Programmering (5%)

Skriv funksjonen **event_results** som tar inn-parametrene **event** og **results**. Parameteren event er en streng som angir én bestemt øvelse (f.eks. *Poker*), mens results vil være datastrukturen som returneres av funksjonen **str_to_numbers** (Oppgave 3e). Funksjonen **event_results** skal returnere en to-dimensonal liste, slik som i eksemplet nedenfor. Listen skal være sortert etter resultat slik at vinneren av den angitte øvelsen og hans/hennes resultat skal kommer først i listen, mens taperen og hans/hennes resultat kommer sist.

Du må gjerne bruke den eksisterende funksjonen **sort_list** hvis du ønsker det. Den står beskrevet i starten av oppgave 3, men oppsummert sorterer den en todimensjonal liste list på elementnummer elem:

```
>>> sort_list([['John', 8], ['Lisa', 12], ['Per', 8]],1)
[['Lisa', 12], ['John', 8], ['Per', 8]]


>>> results # Fra oppgave 3e
[['Name', 'Poker', 'Highjump', 'Balloonshooting', 'SausageEating', 'HoldBreath'], ['John', 8, 1.67, 17, 23, 121.65], ['Lisa', 12, 1.3, 12, 13, 73.02], ['Per', 8, 1.55, 8, 0, 111.35], ['Nelly', 2, 1.34, 9, 17, 31.18], ['Nora', 5, 1.87, 13, 5, 121.65]]

>>> event_results('Poker',results)

[['Lisa', 12], ['John', 8], ['Per', 8], ['Nora', 5], ['Nelly', 2]
```

Skriv ditt svar her...

```
1
```

Maks poeng: 10