NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamen i TDT4110 Informasjonsteknologi - grunnkurs

Faglig kontakt under eksamen:	Alf Inge Wang	Mobil: +47 922 89577
	Terje Rydland	Mobil: +47 957 73463
	Yngve Dahl	Mobil: +47 905 27892

Eksamensdato:	**2017-12-12**
Eksamenstid (fra-til):	**09:00 – 13:00**
Hjelpemiddelkode/Tillatte hjelpemidler:	**Godkjent kalkulator**

Annen informasjon:

Oppgavesettet inneholder 4 oppgaver. Det er angitt i prosent hvor mye hver oppgave og hver deloppgave teller ved sensur. Les igjennom hele oppgavesettet før du begynner å løse oppgavene. Disponer tiden godt! Gjør rimelige antagelser der du mener oppgaveteksten er ufullstendig, skriv kort hva du antar.

Svar kort og klart, og skriv tydelig. Er svaret uklart eller lenger enn nødvendig trekker dette ned.

Målform/språk:	**Bokmål**
Antall sider:	**17 (inkl. Forside, svarark og appendiks)**

Innhold:

- Oppgave 1: Flervalgsoppgave (25%)
- Oppgave 2: Kodeforståelse (20%)
- Oppgave 3: Programmering Gjennomsnittsmåling: (30%)
- Oppgave 3: Programmering Tidevann: (25%)
- Appendiks: Nyttige funksjoner
- Svarark til Flervalgsoppgave (2 eksemplarer)

Informasjon om trykking av eksamensoppgave
Originalen er:

1-sidig ☐ 2-sidig ☐
sort/hvit ☐ farger ☐
skal ha flervalgsskjema ☐

Kontrollert av:

1.des 2017	*Guttorm Sindre*
Dato	Sign.

Oppgave 1: Flervalgsoppgave (25%)

Bruk de to vedlagte svarskjemaene for å svare på denne oppgaven (ta vare på det ene selv). Du kan få nytt ark av eksamensvaktene dersom du trenger dette. Kun ett svar er helt riktig. For hvert spørsmål gir korrekt avkryssing 1 poeng. Feil avkryssing eller mer enn ett kryss gir −1/2 poeng. Blankt svar gir 0 poeng. Du får ikke mindre enn 0 poeng totalt på denne oppgaven. Der det er spesielle uttrykk står den engelske oversettelsen i parentes.

1. Hva er Alan Turing kjent for?
    a. Han utformet det matematiske grunnlaget for dagens datamaskiner.
    b. Han var med på å utforme Turing-arkitekturen.
    c. Han lagde verdens første digitale datamaskin.
    d. Han var en av grunnleggerne av IBM.
2. Hva er fordelene med en SSD (Solid State Drive) sammenliknet med en vanlig harddisk?
    a. Den kan lagre mer data.
    b. Den er raskere.
    c. Den kan ikke krasje.
    d. Den er billigere enn andre disker.
3. I hvilken rekkefølge (etter årstall) vil du plassere oppgitte personer fra datahistorien:
    a. Hollerith - Zuse - von Neumann – Engelbart.
    b. Engelbart - Zuse - Hollerith - von Neumann.
    c. Von Neumann - Zuse - Hollerith – Engelbart.
    d. Zuse - Hollerith - Engelbart - von Neumann.
4. En liten del *inni* en mikroprosessor (CPU) blir ofte forkortet PC, hva står dette for da?
    a. Piece of Crap.
    b. Program Counter.
    c. Personal Computer.
    d. Programming Chip.
5. Hva er rekkefølgen av prefixene relatert til lagring fra minst til størst?
    a. mega, tera, giga, peta.
    b. giga, mega, tera, peta.
    c. mega, giga, tera, peta.
    d. giga, mega, peta, tera.
6. Hva blir binærrepresentasjonen av tallverdien 345?
    a. 101011001.
    b. 1101101010.
    c. 110011.
    d. 11001110.
7. CD-lyd er 16 bits, 44100Hz i stereo. Hvor mye lagringsplass trengs for 35 sekunder i dette formatet?
    a. Ca. 1,54 MB.
    b. Ca. 3,09 MB.
    c. Ca. 6,16 MB.
    d. Oppgaven inneholder ikke tilstrekkelig data til å regne det ut.

8. Hvordan representeres et flyttall i en datamaskin?
    a. Det lagres i maskinen flyttallsminne.
    b. Man avrunder til nærmeste heltall.
    c. Man må konvertere til en tilnærmet verdi som kan representeres.
    d. Man kan ikke med akseptabel nøyaktighet representere et flyttall i en datamaskin.
9. Hvor mange ulike verdier kan representeres med 10 bits?
    a. 512.
    b. 1024.
    c. 1152.
    d. 1280.
10. Hva representerer en bit?
    a. Én bit er for lite til å representere noe.
    b. En farge i RGB-modellen.
    c. En vilkårlig verdi mellom 0 og 1.
    d. To ulike tilstander, men tolkningen av disse er opp til oss.
11. Et lokalt datanettverk (LAN) består av 9 datamaskiner. Nettverket er organisert som en mesh-topologi. Hvor mange direkte forbindelser er det mellom datamaskinene i nettverket?
    a. 9.
    b. 18.
    c. 27.
    d. 36.
12. Hva brukes en Media Access Control (MAC)-adresse til?
    a. Identifisere en spesifikk datamaskin på Internett.
    b. Identifisere en spesifikk datamaskin på et lokalt datanettverk (LAN).
    c. Opprettholde en oversikt over hvilke brukere som skal ha tilgang til hvilke filer.
    d. Varsle sikkerhetsansvarlig om hendelser som kan utgjøre en sikkerhetsrisiko på et lokalt datanettverk (LAN).
13. En organisasjon har fire ulike fysiske datanettverk. Hvorfor kan det være mer hensiktsmessig for organisasjonen å bruke tre rutere fremfor bare én for å knytte nettverkene sammen?
    a. Tre rutere gir alltid bedre nettverkssikkerhet enn én.
    b. Noen av nettverkene kan fremdeles utveksle data selv om én av ruterne skulle slutte å virke.
    c. Én ruter kan kun knytte sammen to nettverk.
    d. Tre rutere gjør at datatrafikken mellom nettverkene kan gå tre ganger så raskt.
14. Hvilken påstand er riktig med hensyn til lag 3 (IP) i TCP/IP-stakken?
    a. Laget er ansvarlig for å merke datapakker med MAC-adresser.
    b. Laget spesifiserer prosedyrer som sikrer pålitelig overføring av data på et nettverk.
    c. Laget spesifiserer formatet på pakker som skal sendes over Internett i tillegg til mekanismer som brukes til å videresende datapakker fra en datamaskin, via en eller flere rutere, og frem til endelig destinasjon.
    d. Både påstand b og c.

15. Hvilke av følgende alternativer er IKKE et sikkerhetsangrep?
    a. Hashing.
    b. Buffer overflow.
    c. SYN flood.
    d. Wiretapping.

16. Hva er pseudokode?
    a. Kode som kan kjøres direkte på prosessoren (CPUen).
    b. Grafisk kode som beskriver en algoritme ved hjelp av ovaler, rektangler, parallellogrammer og piler.
    c. Kode som er basert på kode skrevet av andre utviklere.
    d. Informativ og kompakt beskrivelse av programmering av en algoritme.

17. Rekursjon betyr at:
    a. En funksjon går i evig løkke.
    b. En funksjon kaller seg selv.
    c. En funksjon blir stadig mer effektiv ettersom kjøretiden blir kortere jo flere ganger funksjonen blir kalt.
    d. En funksjon ikke henter inn eller skriver ut informasjon til bruker.

18. Hvilken påstand er USANN om algoritmene binærsøk og sekvensielt søk?
    a. Binærsøk er normalt mer effektiv for lange lister.
    b. Sekvensielt søk kan være mer effektiv enn binærsøk.
    c. Binærsøk fungerer på alle typer lister.
    d. Sekvensielt søk tar lengre tid jo lengre bak i lista elementet man søker etter er.

19. Gitt navnelista Aron, Berit, Daniel, Frank, Jo, Marianne, Oscar, Eskil, Petter, og Stine. Hvilken søkealgoritme er den beste for å finne et navn i lista?
    a. Binærsøk.
    b. Sekvensielt søk.
    c. Begge vil fungere like godt.
    d. Ingen av algoritmene vil fungere.

20. Hva er kjøretids-kompleksiteten til algoritmen LargestNumber som beskrevet under?
    Algorithm LargestNumber

```
Input: A list of numbers L.
   Output: The largest number in the list L.
   if L.size = 0 return null
   largest ← L[0]
   for each item in L, do
        if item > largest, then
            largest ← item
   return largest
```

    Svaralternativer:
    a. Θ(0).
    b. Θ(1).
    c. Θ(n).
    d. Θ(n log n).

Oppgave 2 Kodeforståelse (20%)

Oppgave 2a (5%)

Funksjonen `bin_search` er ment til å skulle utføre binærsøk, men resulterer i feilmeldingen ”`IndexError: list index out of range`”.
I hvilken linje er feilen? (2%)
Hva skulle det egentlig stått på den linjen for at funksjonen skal virke etter sin hensikt? (3%)

```
1     def bin_search(liste, verdi, imin, imax):
2         if(imax < imin):
3             return False
4         else:
5             imid = (imin+imax)
6             if (verdi<liste[imid]):
7                 return bin_search(liste,verdi,imin,imid-1)
8             elif (verdi>liste[imid]):
9                 return bin_search(liste,verdi,imid+1,imax)
10            else:
11                return imid
```

Oppgave 2b (5%)

Hva blir returnert hvis `myst([1,2,3,5,7,9])` med kode som vist under blir kjørt? (3%)
Forklar med en setning hva funksjonen `myst` gjør? (2 %)

```
def myst(A):
     L=len(A)-1
     for i in range(len(A)//2):
          t=A[i]
          A[i] = A[L-i]
          A[L-i]=t
     return A
```

Oppgave 2c (5%)

Hva returneres ved kjøring av funksjonen `myst2(345)` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst2` gjør? (2 %)

```
def myst2(a):
     b=''
     while a or b=='':
          b=str(a%2)+b
          a=a//2
     return b
```

Oppgave 2d (5%)
Forklar med en setning hva funksjonen `myst3([1,2,3,4,5,6,7,8,9,10])` med kode som vist under gjør? (5%)

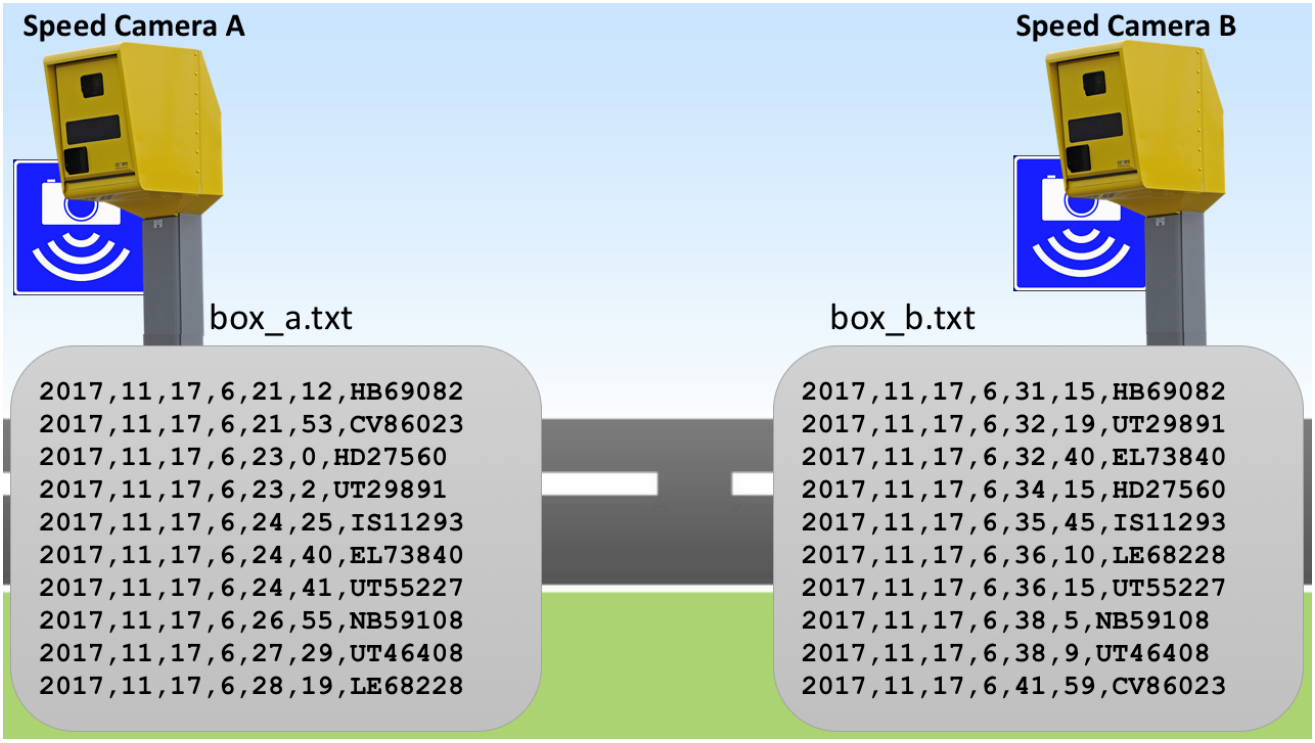
```
import random

def myst3(a):
     b =[0]*len(a)
     for c in range(len(a)):
          d = random.randint(0,len(a)-1)
          b[c] = a[d]
          del a[d]
     return b
```

Oppgave 3 Programmering Gjennomsnittsmåling (30%)

Du kan anta at alle funksjonene mottar gyldige argumenter, og at filer alltid lar seg åpne. Du kan benytte deg av funksjoner fra andre deloppgaver selv om du ikke har løst de deloppgavene.

I denne oppgaven skal du hjelpe politiet med å lage programvare for to fotobokser (fotoboks A og fotoboks B) som blant annet kan brukes til å måle gjennomsnittsfart på kjøretøy. Hver fotoboks gjenkjenner registreringsnummer, dato (år, måned, dag) og tidspunkt (timer, minutter, sekund) for alle biler som passerer i en retning (først fotoboks A, deretter fotoboks B) og lagrer dette i to tekstfiler 'box_a.txt' og 'box_b.txt' (se figuren under).



Oppgave 3a (6%)

Skriv funksjonen `file_to_table` som har en parameter `filename`. Denne funksjonen skal lese inn en tekstfil `filename` fra en fotoboks, som inneholder dato, tid og bilnummer for hver passering. Hver linje i tekstfila er formatert som vist i figuren ovenfor, der første linje har registrert passering av bil med registreringsnummer HB69082 som passerte 17.november 2017 klokken 6:21:12.

Funksjonen skal returnere en todimensjonal tabell (liste av lister), der hver linje inneholder dato, tid og bilnummer. Dato skal oppgis med år, måned og dag som heltall. Tid skal oppgis med time, minutter og sekunder som heltall. Registreringsnummer skal være av typen tekststreng.

Eksempel på kjøring av funksjon på fila box_a.txt som vist i figuren:

```
>>> table = file_to_table('box_a.txt')
>>> print(table)
[[2017, 11, 17, 6, 21, 12, 'HB69082'], [2017, 11, 17, 6, 21, 53, 'CV86023'],
[2017, 11, 17, 6, 23, 0, 'HD27560'], [2017, 11, 17, 6, 23, 2, 'UT29891'], [2017,
11, 17, 6, 24, 25, 'IS11293'], [2017, 11, 17, 6, 24, 40, 'EL73840'], [2017, 11,
17, 6, 24, 41, 'UT55227'], [2017, 11, 17, 6, 26, 55, 'NB59108'], [2017, 11, 17,
6, 27, 29, 'UT46408'], [2017, 11, 17, 6, 28, 19, 'LE68228']]
```

Oppgave 3b (3%)
Skriv funksjonen `time_diff` som tar inn to lister (`start` og `end`), der hver liste beskriver et tidspunkt angitt med dato og klokkeslett. Den første lista (`start`) er tidspunktet for passering av fotoboks A, mens den andre liste (`end`) er tidspunkt for passering av fotoboks B (senere tidspunkt enn A). Funksjonen skal returnere differansen mellom tidspunktene angitt i sekunder. Funksjonen skal også fungere for forskjellige datoer for å ta hensyn til kjøring rundt midnatt. For å beregne antall dager mellom datoer, kan du bruke funksjonen `diff_date(d1,d2)` som returnerer antall dager mellom d2 og d1, der d1 og d2 er dato spesifisert som ei liste på formatet [y,m,d], for eksempel [2017,11,17].

Eksempel på kjøring for å finne differansen i sekunder på passering for bil som passerte fotoboks A 6:24:40 den 17. november 2017 og fotoboks B 6:32:40 samme dag, og et eksempel på en bil som passerte fotoboks A 23:59:59 17. november og fotoboks B 00:09:12 18. november 2017:

```
>>> diff = time_diff([2017,11,17,6,24,40],[2017,11,17,6,32,40])
>>> print(diff)
480
>>> diff = time_diff([2017,11,17,23,59,59],[2017,11,18,0,9,12])
>>> print(diff)
553
```

Oppgave 3c (5%)
Skriv funksjonen `check_min_distance` som har parameterne `car_table` og `diff`. Parameteren `car_table` er en todimensjonal tabell av passeringer som spesifisert i oppgave 3a), mens `diff` er avstand som kreves mellom biler angitt i sekunder. Funksjonen skal returnere registreringsnummer på alle biler som har mindre avstand i sekunder til bilen foran enn `diff`.

Eksempel på kall av funksjonen `check_min_distance` med passeringer fra fotoboks A og for avstand mellom biler på mindre enn 3 sekunder:

```
>>> car_table=file_to_table('box_a.txt')
>>> crazy_drivers=check_min_distance(car_table,3)
>>> print(crazy_drivers)
['UT29891', 'UT55227']
```

Oppgave 3d (4%)
Skriv funksjonen `list_el_cars` som har en parameter `car_table` som er en todimensjonal tabell av passeringer som spesifisert i oppgave 3a). Funksjonen skal returnere antall el-biler som har passert. El-biler har registreringsnummer som starter med EK, EL eller EV.

Eksempel på kall av funksjonen `list_el_cars` med passeringer fra fotoboks A:

```
>>> car_table=file_to_table('box_a.txt')
>>> el_cars=list_el_cars(car_table)
>>> print(el_cars)
1
```

Oppgave 3e (5%)
Skriv funksjonen `generate_license_numbers` som har en parameter `amount`. Denne funksjonen skal returnere ei liste av `amount` antall unike vilkårlige registreringsnummer som kan brukes til testing av systemet. Bokstavene i registreringsnummeret kan være en av følgende: BS, CV, EL, FY, KU, LE, NB, PC, SY, og WC. Tallet i registreringsnummeret skal være mellom 10000 og 99999.

Eksempel på kall av funksjonen `generate_license_numbers`:

```
>>> cars=generate_license_numbers(10)
>>> print(cars)
['CV29728', 'KU73709', 'EL87605', 'NB76581', 'KU65980', 'WC71086', 'PC48338',
'KU36868', 'NB44194', 'KU10621']
```

Oppgave 3f (7%)
Skriv funksjonen `list_speeders` som har fire parameterere `filename_a`, `filename_b`, `speed_limit` og `distance`. De to første parameterne er filnavn på filer som henholdsvis fotoboks A og B har skrevet ut som inneholder passeringer av biler angitt med dato, tidspunkt og registreringsnummer som angitt i oppgave 3a). Parameteren `speed_limit` angir fartsgrensen for strekningen oppgitt i km/t, mens parameteren `distance` oppgir avstand mellom fotoboksene angitt i km. Funksjonen skal returnere en liste av registreringsnummer til alle biler som har kjørt over fartsgrensa (`speed_limit`) for angitt strekning (`distance`). Det vil i praksis si at funksjonen skal liste opp registreringsnummer til alle biler som har brukt for kort tid mellom fotoboksene.

Eksempel på kall av funksjonen `list_speeders` med filene som beskrevet i oppgave 3a) , med fartsgrense 60km/t og distanse på 10km:

```
>>> speeders = list_speeders('box_a.txt','box_b.txt',60,10)
>>> print(speeders)
['UT29891', 'EL73840', 'LE68228']
```

Oppgave 4 Programmering Tidevann (25%)

I denne oppgaven skal vi se på tidspunktene for tidevann i Trondheim. Tidevann er varierende vannstand som forårsakes av solens og månens påvirkning på Jorda. Det er høyvann når man har maksimal vannstand, mens ved lavvann har man laveste vannstand. Det går 12 timer, 25 minutter og 12 sekunder mellom to høyvann. Midt mellom høyvann er det lavvann.

Som et eksempel: dersom det er høyvann klokka 00:00:00, så er det også høyvann klokka 12:25:12, samt at det er lavvann klokka 06:12:36 og klokka 18:37:48.

Vi skal i denne oppgaven begrense oss til tidevannet i Trondheim i desember 2018. Det første tidevannet i byen denne måneden er 1. desember 2018 kl. 03:18, og det var lavvann.

Oppgave 4a (3%)

Skriv funksjonen `formatTime` som har en parameter `seconds` som er antall sekunder som har gått siden midnatt. Funksjonen skal returnere en streng som inneholder klokkeslettet på formatet `hh:mm:ss`. Både timer, minutter og sekunder skal skrives med to siffer, og det skal brukes innledende null brukes dersom det er nødvendig. Funksjonen trenger ikke å håndtere verdier over 86400 sekunder (dvs. over ett døgn).

Eksempel på kall av funksjonen `formatTime`:

```
>>> time = formatTime(12305)
>>> print(time)
03:25:05
```

Oppgave 4b (2%)

Skriv en funksjon `valuesDecember` som har null parametere, men som skal returnere to konstanter `first` og `period`. Den første returverdien (`first`) angir tidspunktet for første lavvann som skal være antall sekunder tidspunktet klokka 03:18 er siden midnatt natt til 1.desember. Den andre returverdien (`period`) angir tiden i antall sekunder mellom to høyvann (eller to lavvann) for desember måned som er satt til å være 12 timer, 25 minutter og 12 sekunder.

Eksempel på kall av funksjonen `valuesDecember`:

```
>>> first, period = valuesDecember()
>>> print(first)
11880
>>> print(period)
44712
```

Oppgave 4c (5%)
Skriv en funksjon `genTides` som har ingen parametere, men skal benytte seg av funksjonen i 4b) for å finne verdier for desember måned. Funksjonen `genTides` skal returnere to ulike returverdier. Den første er en liste med tidspunktene for alle lavvann i desember, der tidspunktene er angitt som antall sekunder siden midnatt på natten til 1. desember. Den andre returverdien er en tilsvarende liste med tidspunktene for alle høyvann for samme måned – i samme format. Det er 31 dager i desember.

Eksempel på kall av funksjonen `genTides` og utskrift av de åtte første elementene av resultatet:

```
>>> lows,highs=genTides()
>>> print(lows[:8])
[11880, 56592, 101304, 146016, 190728, 235440, 280152, 324864]
>>> print(highs[:8])
[34236, 78948, 123660, 168372, 213084, 257796, 302508, 347220]
```

Oppgave 4d (3%)
Skriv en funksjon `genTidesStr` som har parameteren `tideList` som er ei liste av tidspunkter angitt i antall sekunder siden starten av måneden på likt format med hva som ble returnert i oppgave 4c). Funksjonen skal returnere ei liste av tekststrenger, der hver tekststreng inneholder først tallet for dagen i måneden og deretter klokkeslett formatert angitt med timer, minutter og sekunder.

Eksempel på kall av funksjonen `genTidesStr` og utskrift av de fem første elementene av resultatet:

```
>>> lows, highs = genTides()
>>> lowStrings = genTidesStr(lows)
>>> for item in lowStrings[:5]:
      print(item)

1 03:18:00
1 15:43:12
2 04:08:24
2 16:33:36
3 04:58:48
```

Oppgave 4e (7%)
Skriv en funksjon `checkTides` som har parameteren `dayInMonth` som er et heltall. Funksjonen sjekker om det er høyvann eller lavvann i eksamenstiden på denne dagen i måneden, det vil si mellom klokken 09:00 og 13:00. Funksjonen skal skrive ut en streng på ett av følgende format: ’no tides’, ’high tide at 09:10:11’ eller ’low tide at 12:13:14’. De faktiske klokkeslettene skal stemme overens med dataene som funksjonen `genTides` returnerer.

Eksempel på kall av funksjonen `checkTides` for 12., 18. og 24. desember:

```
>>> checkTides(12)
no tides
>>> checkTides(18)
high tide at 12:12:36
>>> checkTides(24)
low tide at 11:02:24
```

Oppgave 4f (5%)
Skriv en funksjon `listTides` som ikke har noen parametere og skal heller ikke returnere noe. Funksjonen skal skrive ut alle lavvann for desember måned 2018 i en tabell, slik at alle lavvann på samme dato listes på samme linje i den rekkefølgen de kommer. Funksjonen skal hente data fra funksjonen `genTides`. Utskriften skal skrives ut i tre kolonner: første kolonne er dagen i måneden, andre kolonne er tidspunkt for dagens første lavvann, og tredje kolonne er tidspunkt for et eventuelt andre lavvann den dagen. Alle data skal være satt opp og justert under hverandre som vist i eksemplet på kjøring under.

Eksempel på kall av funksjonen `listTides` for desember:

```
>>> listTides()
Day  First    Second
  1 03:18:00 15:43:12
  2 04:08:24 16:33:36
  3 04:58:48 17:24:00
  4 05:49:12 18:14:24
  5 06:39:36 19:04:48
  6 07:30:00 19:55:12
  7 08:20:24 20:45:36
  8 09:10:48 21:36:00
  9 10:01:12 22:26:24
 10 10:51:36 23:16:48
 11 11:42:00
 12 00:07:12 12:32:24
 13 00:57:36 13:22:48
 14 01:48:00 14:13:12
 15 02:38:24 15:03:36
 16 03:28:48 15:54:00
 17 04:19:12 16:44:24
 18 05:09:36 17:34:48
 19 06:00:00 18:25:12
 20 06:50:24 19:15:36
 21 07:40:48 20:06:00
 22 08:31:12 20:56:24
 23 09:21:36 21:46:48
 24 10:12:00 22:37:12
 25 11:02:24 23:27:36
 26 11:52:48
 27 00:18:00 12:43:12
 28 01:08:24 13:33:36
 29 01:58:48 14:24:00
 30 02:49:12 15:14:24
 31 03:39:36 16:04:48
```

Appendix: Useful Functions and Methods

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.

//
: Floor/integer division: Returns the integral part of the quotient.

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.

String methods:

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.center(width)
: Return the string center justified in a string of length width.

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.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 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).

Random methods:

random.random()
: Return the next random floating point number in the range [0.0, 1.0).

random.randint(a,b)
: Return a random integer *N* such that a <= N <= b.

random.choice(seq)
: Return a random element from the non-empty sequence seq. If seq is empty, raises IndexError.

random.randrange(start, stop [, step])
: Return a randomly selected element from range(start, stop, step).

List operations:

s[i:j:k]
: Return slice starting at position i extending to position j in k steps. 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.

Files

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).

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 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.close()
: Close the file and free up any system resources taken up by the open file.

Svarskjema flervalgsoppgave

Kandidatnummer: _____ Program: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgavenr	***A***	***B***	***C***	***D***
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				

Denne siden er med hensikt blank!

Svarskjema flervalgsoppgave

Kandidatnummer: _____ Program: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgavenr	***A***	***B***	***C***	***D***
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				