NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Kontinuasjonseksamen i TDT4110 Informasjonsteknologi - grunnkurs

Faglig kontakt under eksamen: Alf Inge Wang Mobil: +47 922 89577
Guttorm Sindre Mobil: +47 944 30245

Eksamensdato: **2017-08-XX**

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: Programmering Priskrig: (25%)
- Oppgave 3: Programmering Storskjerm: (30%)
- Oppgave 4: Kodeforståelse (20%)
- Appendiks: Nyttige funksjoner
- Svarark til Flervalgsoppgave (2 eksemplarer)

Informasjon om trykking av eksamensoppgave
Originalen er:
1-sidig ☐ 2-sidig ☐
sort/hvit ☐ farger ☐

Kontrollert av:

27 juni 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. Hvor mange *bytes* trenger man for å representere et full-HD bilde (1920x1080) i sort/hvitt?
   a. 86 400.
   b. 207 360.
   c. 259 200.
   d. 2 073 600.

2. Hva kalles kretskortet i en PC som knytter sammen CPU, minnet, grafikkort og annen tilleggsfunksjonalitet?
   a. PC-kort.
   b. Hovedkort (motherboard).
   c. Flerkjernekort (Multi Core Board).
   d. Datterkort.

3. Hva ligger i begrepet Random Access Memory (RAM)?
   a. Data hentes/skrives direkte uavhengig hvor det ligger i minne.
   b. Data hentes/skrives sekvensielt i minne.
   c. Det er tilfeldig hvilke enheter som har tilgang til ulike deler av minne.
   d. Hastigheten på lasting/skriving av data i minne er tilfeldig.

4. Hva er hovedforskjellen på Primær- og Sekundærminne?
   a. Sekundærminne er alltid raskere enn Primærminne.
   b. Sekundærminne fungerer som backup hvis Primærminne slutter å fungere.
   c. Sekundærminnet er permanen, mens primærminnet er flytig.
   d. Primærminne er billigere per Megabyte enn Sekundærminne.

5. Hvilken påstand er IKKE KORREKT om fotolitografi?
   a. Brukes til å fremstille integrerte kretser (IC).
   b. Kostnaden og mengde arbeid er den samme uavhengig av hvor komplisert kablingen er.
   c. Prosessen åpner for å legge flere lag med kretser oppå hverandre.
   d. RGB benyttes for å eksponere ulike lag som for eksempel fotoresist (blå), ubeskyttet metall (grønn) og andre lag (rød).

6. Hvilken beskrivelse passer best på en transistor?
   a. Konverterer fra analogt til digitalt signal.
   b. Konverterer fra digitalt til analogt signal.
   c. Fungerer som en bryter som styres ved hjelp av påført strøm.
   d. Overfører et digitalt signal fra en fysisk enhet til en annen.

7. Hvilket alternativ beskriver ”Fetch/Execute Cycle” best?
   Forkortelser i alfabetisk rekkefølge: DF = Data Fetch, EX = Instruction Execution, IF = Instruction Fetch, ID = Instruction Decode, RR = Result Return
   a. IF, ID, DF, EX, RR
   b. DF, IF, ID, EX, RR
   c. IF, ID, EX, DF, RR
   d. RR, DF, IF, ID, EX

8. Hva er hovedoppgaven til ALU?
   a. Sørge for å hente og utføre instruksjoner.
   b. Knytte sammen input og output enheter.
   c. Utføre regneoperasjoner.
   d. Styre programtelleren (Program Counter).

9. Hva blir resultatet av binæraddisjon av 10101+10101?
   a. 101010.
   b. 101000.
   c. 100000.
   d. 110001.

10. Hva er det binære tallet som tilsvarer det heksadesimale tallet D020?
    a. `1101 0010`.
    b. `110 101 010 000`.
    c. `0000 0010 0000 1101`.
    d. `1101 0000 0010 0000`.

11. Hvis ’OSTE’ kodet i ASCII blir ’`0100 1111 0101 0011 0101 0100 0100 0101`’, hva blir ’POP’ kodet i ASCII?
    a. ’`0100 1111 0101 0011 0101 0100`’.
    b. ’`0101 0011 0101 0100 0100 0101`’.
    c. ’`0100 1111 0101 0000 0101 0000`’.
    d. ’`0101 0000 0100 1111 0101 0000`’.

12. Hva sier Nyquist-regelen om digitalt lydopptak?
    a. Samplingsfrekvensen må være halvparten så rask som den raskeste lydfrekvensen.
    b. Samplingsfrekvensen må være minst dobbelt så rask som den raskeste lydfrekvensen.
    c. Samplingsfrekvensen må være den samme som den raskeste lydfrekvensen.
    d. Samplingsfrekvensen må være på 4410Hz.

13. Hvilken påstand er IKKE KORREKT om JPEG?
    a. Bildefiler i JPEG-format er mindre i størrelse enn ikke-komprimerte bildefiler.
    b. JPEG-formatet bruker komprimering med tap av bildekvalitet.
    c. JPEG-formatet egner seg best for bilder med enkel datagrafikk.
    d. Det er en direkte sammenheng mellom bildekvalitet og komprimering.

14. Hva står forkortelsen ISP i pensumboka for?
    a. Internal Storage Protocol.
    b. Internet Service Provider.
    c. Integrated Software Process.
    d. Illustrated Software Plan.

15. I hvilket lag i TCP/IP referansemodellen finner man HTTP, SMTP, og FTP?
    a. Applikasjonslaget.
    b. Transportlaget.
    c. Nettverkslaget.
    d. Linklaget.

16. Hva er en pakke (packet) i nettverkssammenheng?
    a. En datablokk av fast lengde som sendes gjennom nettverket, fra avsender til mottaker.
    b. En datamelding med varierende lengde som inneholder all data som sendes fra avsender til mottaker.
    c. En fil som blir komprimert før den sendes over nettverk til mottaker.
    d. En kryptert fil som sendes over nettverk som må pakkes opp før den kan brukes hos mottaker.

17. Hva er en protokoll i nettverkssammenheng?
    a. En avtale mellom nettverkseier og en bedrift som bruker nettet.
    b. Et register der all nettverkstrafikk blir lagret for gjennomsyn av myndigheter.
    c. Et sett av kommunikasjonsregler for utveksling av data.
    d. En komprimeringsalgoritme som gjør det mer effektivt å sende data over nett.

18. Hva er algoritmekompleksiteten til binærsøk?
    a. $\Theta(\log n)$.
    b. $\Theta(n)$.
    c. $\Theta(n \log n)$.
    d. $\Theta(n^2)$.

19. Hva er algoritmekompleksiteten til "brute force" (travelling salesman problemet)?
    a. $\Theta(n^2)$
    b. $\Theta(n^3)$
    c. $\Theta(2^n)$
    d. $\Theta(n!)$

20. Hva er en ulempe med inkrementell utvikling innen systemutvikling?
    a. Vanskelig å håndtere endringer underveis.
    b. Alle krav må være spesifiserte på forhånd.
    c. Vanskeligere for prosjektledere å styre leveranser for å måle framdrift.
    d. Fungerer kun for store prosjekter.

Oppgave 2 Programmering Priskrig (25%)

pricewar.txt

Rema	Milk	14.50
Rema	Pepsi Max	20.00
Extra	Milk	14.20
Kiwi	Pepsi Max	20.50
Extra	Pepsi Max	19.50
Rema	Banana	12.50
Kiwi	Milk	13.00
Rema	Juice	29.30
Extra	Juice	23.00
Rema	Chocolate	14.00
Extra	Chocolate	13.30
Kiwi	Chocolate	13.00
Kiwi	Banana	10.50
Extra	Banana	11.00

Du kan anta at alle funksjonene mottar gyldige argumenter (inn-verdier). Du kan benytte deg av funksjoner fra deloppgaver selv om du ikke har løst deloppgaven.

I denne oppgaven skal du lage et program for å sammenlikne priser på utvalgte varer fra forskjellige butikker[1]. Utgangspunktet for denne sammenlikningen er ei tekstfil der hver linje består av tre elementer adskilt med tabulator (”`\t`”): Navn på butikkjede, Navn på vare, og Pris (se tekstboks). Merk at en slik tekst fil kan ha varierende antall butikkjeder, samt varierende antall varer som sammenliknes.

Oppgave 2a (5%)
Skriv funksjonen `file_to_list` som har en input-parameter `filename`. Denne funksjonen skal lese inn en tekstfil `filename` og returnere en tabell (liste av lister), der hver rekke inneholder navn på butikkjede, navn på vare, og pris på vare. Merk at pris på vare skal representeres som et flyttall (float).

Eksempel på kall av funksjon med filen ’pricewar.txt’ som vist ovenfor:

```
>>> dataList = file_to_list('pricewar.txt')
>>> print(dataList)
[['Rema', 'Milk', 14.5], ['Rema', 'Pepsi Max', 20.0], ['Extra', 'Milk', 14.2],
['Kiwi', 'Pepsi Max', 20.5], ['Extra', 'Pepsi Max', 19.5], ['Rema', 'Banana',
12.5], ['Kiwi', 'Milk', 13.0], ['Rema', 'Juice', 29.3], ['Extra', 'Juice', 23.0],
['Rema', 'Chocolate', 14.0], ['Extra', 'Chocolate', 13.3], ['Kiwi', 'Chocolate',
13.0], ['Kiwi', 'Banana', 10.5], ['Extra', 'Banana', 11.0], ['Kiwi', 'Juice',
27.5], ['Bunnpris', 'Milk', 13.0], ['Bunnpris', 'Pepsi Max', 21.5], ['Bunnpris',
'Banana', 15.9], ['Bunnpris', 'Juice', 26.0], ['Bunnpris', 'Chocolate', 12.5]]
>>>
```

Oppgave 2b (4%)
Skriv funksjonen `list_stores` som har `dataList` som input-parameter. `dataList` er en tabell (liste av lister) lik den som blir returnert fra funksjonen `file_to_list` i Oppgave 2a. Funksjonen skal returnere en komplett liste av butikkjeder den finner i tabellen `dataList`. Hver butikkjede skal kun ha ett innslag i lista. Merk også at man aldri vet hvilke butikkjeder som lista vil inneholde. Rekkefølgen på butikkjedene skal samsvare med rekkefølgen de kommer i tabellen `dataList`.

Eksempel på kall av funksjon med filen ’pricewar.txt’ som vist ovenfor:

```
>>> dataList = file_to_list('pricewar.txt')
>>> storeList = list_stores(dataList)
>>> print(storeList)
['Rema', 'Extra', 'Kiwi', 'Bunnpris']
>>>
```

Oppgave 2c (5%)

[1] De oppgitte prisene er fiktive og ikke reelle priser fra de oppgitte butikkjedene

Skriv funksjonen `sum_prices_stores` som har input-parameterne `dataList` og `storeList` (fra Oppgave 2a og 2b). Funksjonen skal returnere en liste av totalsummen for alle varene per butikkjede. Rekkefølgen på totalsummene skal være den samme som rekkefølgen på butikkjedene i `storeList`.

Eksempel på kall av funksjon med fila 'pricewar.txt' som vist ovenfor. Resultatet er summen av priser for butikkjedene Rema, Extra, Kiwi og Bunnpris (i samme rekkefølgen som Oppgave 2b).

```
>>> dataList = file_to_list('pricewar.txt')
>>> storeList = list_stores(dataList)
>>> sumStores = sum_prices_stores(dataList,storeList)
>>> print(sumStores)
[90.3, 81.0, 84.5, 88.9]
>>>
```

Oppgave 2d (6%)
Skriv funksjonen `rank_stores` som har input-parameterne `storeList` og `sumStores` (fra oppgave 2b og 2c). Funksjonen skal returnere ei liste med navnene til butikkjedene sortert fra kjeden med lavest pris til høyest pris.

Eksempel på kall av funksjon med fila 'pricewar.txt' som vist ovenfor. Merk at før `rank_stores` kjøres, er lista av butikkjeder i samme rekkefølge som i tekstfila 'pricewar.txt'. Etter å ha kjørt funksjonen `rank_stores`, er rekkefølgen sortert etter butikkjeder med lavest pris.

```
>>> dataList = file_to_list('pricewar.txt')
>>> storeList = list_stores(dataList)
>>> print(storeList)
['Rema', 'Extra', 'Kiwi', 'Bunnpris']
>>> sumStores = sum_prices_stores(dataList,storeList)
>>> storeList = rank_stores(storeList,sumStores)
>>> print(storeList)
['Extra', 'Kiwi', 'Bunnpris', 'Rema']
```

Oppgave 2e (5%)
Skriv funksjonen `store_analysis` som har input-parameteren `filename`. Funksjonen skal laste inn ei fil med filnavnet `filename`, og deretter skrive ut summen for varene for hver butikk, og deretter skrive ut ranking av butikkjeder sortert etter der varene i fila `filename` er billigst. Funksjonen skal ikke returnere noe, men ha en utskrift til skjerm som vist under.

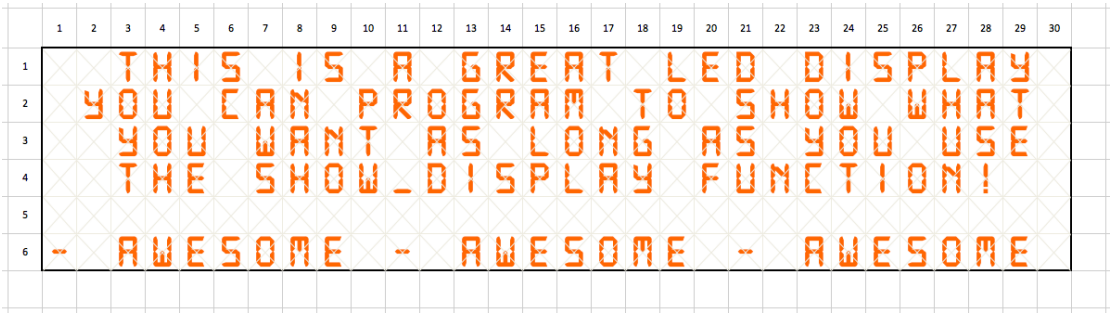
Eksempel på kall av funksjon med fila 'pricewar.txt' som vist ovenfor.

```
>>> store_analysis('pricewar.txt')
The total price for shopping per store is:
Rema : 90.3 kr
Extra : 81.0 kr
Kiwi : 84.5 kr
Bunnpris : 88.9 kr

The ranking of stores according to prices is:
1 Extra
2 Kiwi
3 Bunnpris
4 Rema
>>>
```

Oppgave 3 Programmering Storskjerm (30%)

I denne oppgaven skal du hjelpe *Katpiss Everbeen* til å lage funksjoner som skal brukes til å lage et system for å vise fram tekst på storskjerm ved store arrangementer. Denne storskjermen kan vise 6 linjer med tekst, der hver linje består av 30 tegn eller bokstaver som vist i `Figur 1`.



Figur 1 Storskjerm

Storskjermen kommer med funksjonen `show_display` for å vise fram tekst på skjermen som du kan bruke i din kode. Funksjonen har input-parameteren `content`, som er en liste av seks elementer, der hvert element er en tekststreng på nøyaktig 30 tegn eller bokstaver. Hvis man prøver å kalle funksjonen med en liste med feil dimensjoner, vil ikke noe vises på storskjermen og funksjonen gir feilmeldingen `"Error: Wrong dimensions"`. Storskjermen kan bare vise fram store bokstaver, men funksjonen `show_display` vil selv sørge for å oversette fra små til store bokstaver hvis det trengs.

I denne oppgaven anbefales det å gjenbruke funksjoner fra andre deloppgaver der det er naturlig. Du kan bruke funksjoner fra andre deloppgaver selv om du ikke har løst denne deloppgaven.

Oppgave 3a (4%)

Skriv funksjonen `enter_line` som har to input-parametere `prompt` og `length`. Funksjonen skal spørre brukeren om å skrive inn en setning som skal returneres som en tekststreng. Setningen skal være av lengde spesifisert av input-parameteren `length`. Hvis setningen ikke er av spesifisert lengde, skal funksjonen gi feilmeldingen: ”The text must be `[length]` characters long”, og fortsette å spørre om en ny setning til brukeren har gitt en med korrekt lengde. Parameteren `prompt` spesifiserer hva brukeren skal spørres om.

Eksempel på kall av funksjon (bruker-input er skrevet med **fet font**):

```
>>> enter_line("Enter line 1: ",30)
Enter line 1: ITGK is the best!
The text must be 30 characters long
Enter line 1: This is a test on writing nicely and cooly!
The text must be 30 characters long
Enter line 1: This is a test on writing nice
'This is a test on writing nice'
>>>
```

Oppgave 3b (4%)
Skriv funksjonen `adjust_string` som har to input-parametere `text` og `length`. Funksjonen skal returnere en ny utgave av tekststrengen `text` som har lengde `length`. Hvis strengen `text` har flere tegn enn `length`, skal den resterende teksten kuttes. Hvis strengen `text` har færre tegn enn `length`, skal teksten midtstilles og man skal legge til mellomrom (space) slik at lengden på strengen som returneres blir akkurat `length`.

Eksempel kall av funksjonen `adjust_string` er vist under:

```
>>> adjust_string("This is a test on writing nicely and cooly!",30)
'This is a test on writing nice'
>>> adjust_string("ITGK is the best!",30)
'      ITGK is the best!       '
>>> adjust_string("ITGK",30)
'             ITGK             '
>>>
```

Oppgave 3c (3%)
Skriv en smartere versjon av versjon av funksjonen `enter_line_smart` (fra Oppgave 3a) som har to input-parametere `prompt` og `length`. Funksjonen skal ta imot input fra brukeren ved å bruke spørreteksten `prompt`, og returnere en streng på lengde `length`. Hvis teksten brukeren skriver inn er lengre enn `length` skal resterende teksten kuttes, og hvis teksten brukeren skriver inn er kortere skal teksten midtstilles og fylles ut med mellomrom (space) slik at teksten blir på `length` antall tegn.

Eksempel kall av funksjonen `enter_line_smart` er vist under:

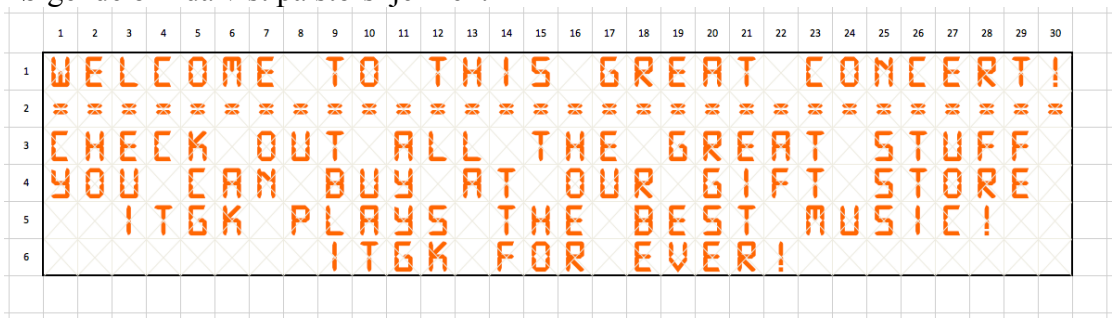
```
>>> enter_line_smart("Enter line 1: ",30)
Enter line 1: ITGK is the best!
'      ITGK is the best!       '
>>> enter_line_smart("Enter line 2: ",30)
Enter line 2: This is a test on writing nicely and cooly!",30)
'This is a test on writing nice'
>>> enter_line_smart("Enter line 3: ",30)
Enter line 3: ITGK
'             ITGK             '
>>>
```

Oppgave 3d (4%)
Skriv funksjonen `enter_show_text` som spør brukeren om å legge inn seks linjer med tekst på 30 tegn, og deretter viser innholdet på storskjermen. Funksjonen har ingen input-parametere og returnerer ingen ting. Hvis teksten som brukeren skriver inn er over 30 tegn, skal overflødig tekst kuttes bort. Hvis teksten som brukeren skriver inn er under 30 tegn, skal teksten midtstilles og fylles ut med mellomrom (space) slik at teksten blir på 30 tegn.

Eksempel på kjøring (bruker-input er skrevet med **fet font**):

```
>>> enter_show_text()
Line 1: Welcome to this great concert!
Line 2: ==============================
Line 3: Check out all the great stuff
Line 4: you can buy at our gift store
Line 5: ITGK plays the best music!
Line 6: ITGK for ever!
```

Følgende blir da vist på storskjermen:



Oppgave 3e (5%)
Skriv funksjonen `scroll_display` som har to input-parametere `content` og `line`. Funksjonen returnerer ingen ting. Parameteren `content` er ei liste bestående av 6 tekststrenger på 30 tegn, og parameteren `line` er et heltall mellom 1 og 6. Funksjonen skal vise fram innholdet fra lista `content` på storskjermen, der teksten på linje `line` skal roteres mot venstre (scrolle) helt til teksten på denne linja er tilbake der den startet (som vist på figurene nederst). Oppdatering av storskjermen skal skje hvert tiendedels sekund (0,1 sek). Teksten på linje `line` vil altså forflytte seg 30 ganger mot venstre før funksjonen avslutter. Du kan anta at funksjonen kalles med riktige argumenter (`content` inneholder 6 strenger på 30 tegn og `line` er heltall mellom 1 og 6). Tidsforsinkelse gjøres ved å bruke funksjonen `sleep(s)` fra biblioteket `time`, der `s` spesifiserer antall sekunder tidsforsinkelse. Eks på bruk: `time.sleep(0.5)` gir en pause på ½ sekund.

Eksempel på kjøring der linje 1 (øverste linja) roteres (scrolles) mot venstre:

```
>>> content=['Welcome to this great concert!',
          '==============================',
          'Check out all the great stuff ',
          'you can buy at our gift store ',
          '  ITGK plays the best music!  ',
          '         ITGK for ever!       ']
>>> scroll_display(content,1)
```

Følgende blir da vist på storskjermen (viser utdrag av hendelsesforløpet):

0,1 sekunder senere:

0,1 sekunder senere:

… og helt til slutt (3 sekunder senere):

Oppgave 3f (10%)
Skriv funksjonen `display_from_file` som har en input-parameter `filename`. Funksjonen skal lese inn fra tekstfila `filename`, og vise fram innholdet i tekstfila på storskjermen seks linjer av gangen. Funksjonen returnerer ingen ting. Hvis teksten på ei linje i fila er over 30 tegn, skal resterende tekst kuttes. Hvis teksten på ei linje i fila er under 30 tegn, skal teksten midtstilles og fylles ut med mellomrom (space). Funksjonen skal ha 10 sekunders pause mellom hver gang nytt innhold vises på skjermen. Man kan anta at fila har et antall linjer som går opp i seks-gangen.

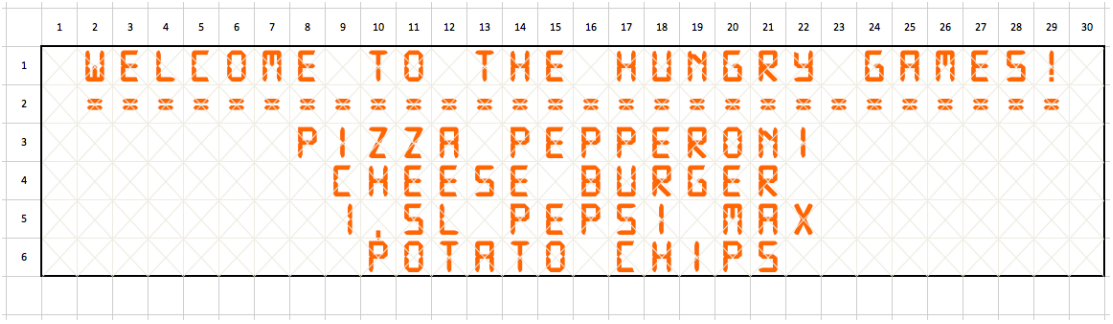
Anta at innholdet på tekstfila `message.txt` er:

```
Welcome to the Hungry Games!
============================
Pizza pepperoni
Cheese burger
1,5L Pepsi Max
Potato chips
You will not get any of
these snacks when competing!
Stay hungry and look at all
the stuff you cannot eat!
Happy Hungry Games, and may
the odds be ever in your favor
```

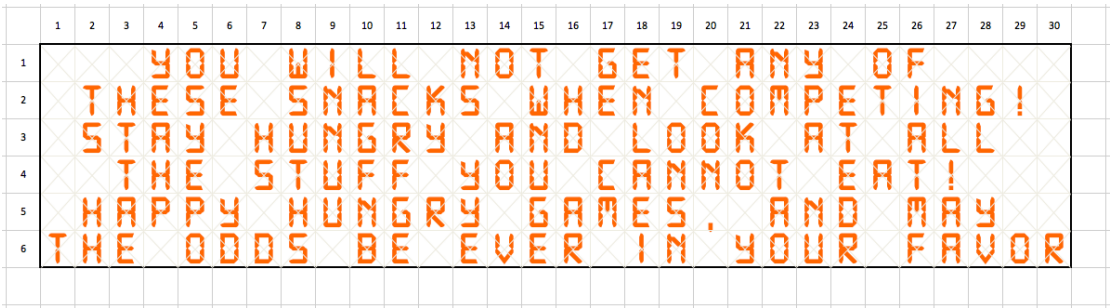
Eksempel på kall av funksjonen med innholdet av `message.txt` som vist over blir:

```
>>> display_from_file('message.txt')
>>>
```

Følgende blir da vist fram på storskjermen:



Ti sekunder senere blir følgende vist fram på storskjermen:



Oppgave 4 Kodeforståelse (20%)

Oppgave 4a (5%)
Hva returneres ved kjøring av funksjonen `myst1('G dg','omd!','dia!')` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst1()` gjør? (2 %)

```
def myst1(s1,s2,s3):
  s = ''
  for i in range(len(s1)):
    s += s1[i]+s2[i]+s3[i]
  return s
```

Oppgave 4b (5%)
Hvilken verdi for m når man kjører koden nedenfor? (3 %)
Forklar med en setning hva funksjonen `myst2( )` gjør? (2 %)

```
def myst2(m):
  for y in range(len(m)):
    for x in range(len(m[0])):
      if y==0 or y==len(m)-1:
        m[y][x] = 0
      elif x==0 or x==len(m)-1:
        m[y][x] = 0
  return m

m=[[1,2,3,4,5],
   [2,3,4,5,6],
   [3,4,5,6,7],
   [4,5,6,7,8],
   [5,6,7,8,9]]

m = myst2(m)
```

Oppgave 4c (5%)
Hva returneres ved kjøring av funksjonen `myst3('xsidrwteasMc hedhfT')` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst3()` gjør? (2 %)

```
def myst3(s):
  a = ''
  for x in range(len(s)-1,-1,-2):
    a+= s[x]
  return a
```

Oppgave 4d (5%)
Hva returneres ved kjøring av funksjonen `myst4(2,1,4)` med kode som vist under? (3%)
Forklar med en setning hva funksjonen `myst4()` gjør? (2%)

```
def myst4(x,y,z):
  if y<z:
    return myst4(x*x,y+1,z)
  else:
    return x
```

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.

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.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.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).

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.

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				