NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Kontinuasjonseksamen i TDT4105 Informasjonsteknologi - grunnkurs

Faglig kontakt under eksamen:	Rune Sætre	Mobil: +47 452 18103
	Anders Christensen	Mobil: +47 918 97181
	Terje Rydland	Mobil: +47 957 73463

Eksamensdato:	**2016-08-12**
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:	**19 (inkl. Forside, svarark og appendiks)**

Innhold:

Kontrollert av:

22.juni 2016 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. En byte inneholder hvor mange bits?
   a. 1
   b. 8
   c. 16
   d. 32

2. Hva kalles en test der man bare ser på inputs og outputs av en funksjon?
   a. Hvitbokstest (white box test).
   b. IO-test (Input Output test).
   c. Svartbokstest (black box test).
   d. Integrasjonstest.

3. Omtrent hvor mange enheter er koblet til Internett i dag?
   a. Halvparten så mange som det finnes mennesker på jorden.
   b. Like mange som det finnes mennesker på jorden.
   c. Over tre ganger så mange som det finnes mennesker på jorden.
   d. Det er det umulig å svare på.

4. Hva er en transistor?
   a. Enheten som omformer 220V vekselstrøm til likestrøm som kan brukes til de ulike enhetene (CPU, lydkort, grafikkort, harddisk, RAM osv.) i datamaskinen.
   b. Et kretskort der man kobler samme de ulike delene i en datamaskin, som CPU, minne, lydkort, grafikkort, RAM osv.
   c. En bryter som det kan enten gå strøm igjennom eller ikke, som man kan endre ved hjelp av strøm.
   d. Algoritmen fra å gjøre om fysiske lydsignaler til digital representasjon av 0er og 1ere.

5. Hvordan representeres desimaltallet 321 som et binært tall?
   a. `1100101`
   b. `11001001`
   c. `100101101`
   d. `101000001`

6. Hva er hensikten med en DAC?
   a. Konvertere fra analogt til digitalt signal.
   b. Konvertere fra digitalt til analogt signal.
   c. Komprimere et digitalt signal.
   d. Øke samplingsraten.

7. Hva får vi med like mengder av rødt, grønt og blått i et punkt på en skjerm?
   a. Sort, hvitt eller gråtoner.
   b. Cyan, magenta og gult.
   c. Brunt.
   d. Fiolett.

8. Hva menes med 'sampling rate' under digitalisering av lyd?
   a. Hvor hyppig man måler lyden.
   b. Frekvensen på lyden som skal digitaliseres.
   c. Nøyaktigheten/antall bits i måleverdien.
   d. Den maksimale lydstyrken/volumet som kan digitaliseres.

9. Hva er RGB?
   a. Random GB - hukommelsen i en datamaskin.
   b. En fargemodell for reflektert lys som viser hvordan alle farger dannes fra primærfargene.
   c. En fargemodell for utstrålt lys som viser hvordan alle farger dannes fra primærfargene.
   d. Rødt, Gult og Blått - primærfargene i reflektert lys.

10. Hva er «booting» av en datamaskin?
    a. Prosessen som starter opp datamaskinen.
    b. Endre systeminnstillinger på en datamaskin.
    c. Sjekke om datamaskinen er smittet av et virus (en boot).
    d. Sparke til en datamaskin når den ikke gjør det man ønsker.

11. Hva ligger i begrepet" system engineering" ifølge læreboka?
    a. "System engineering" fokuserer på underliggende teorier og metoder som utgjør datasystemer.
    b. "System engineering" fokuserer på praktiske problemer med å produsere programvare.
    c. "System engineering" inkluderer alle aspekter av utvikling og evolusjon av komplekse systemer hvor programvare spiller en viktig rolle.
    d. "System engineering" og "Software engineering" er det samme.

12. Hvilke av disse enhetene er vanligvis involvert i "Fetch/Execute Cycle"?
    a. ALU, CU (Control Unit), RAM.
    b. ROM, ALU, RAM.
    c. CU, RAM, ROM.
    d. OS, ALU, CU.

13. Hva går aktiviteten arkitekturdesign (architectural design) ut på?
    a. Definere grensesnitt mellom systemkomponenter.
    b. Designe hvordan hver systemkomponent skal fungere.
    c. Designe datastrukturene for hele systemet.
    d. Identifisere den overordnete strukturen av system, de overordnete komponentene, og hvordan de er strukturert og knyttet til hverandre.

14. Når det gjelder henholdsvis lagring og overføring av data så måles de vanligvis i …
    a. Bits (lagring) og Bits pr. sekund (overføring).
    b. Bits (lagring) og Byte pr. sekund (overføring).
    c. Byte (lagring) og Byte pr. sekund (overføring).
    d. Byte (lagring) og Bits pr. sekund (overføring).

15. En ulempe med inkrementell utvikling kan være:
    a. Må ha alle krav på plass før man kan starte på design og implementering av system.
    b. Gjør det vanskeligere å teste systemet for feil.
    c. Egner seg kun for store prosjekter.
    d. Arkitekturen (strukturen) til systemet har en tendens til å forringes for hvert inkrement.

16. Man kan sende stadig mer informasjon gjennom luften ved å …
    a. Bruke eldre stabile omkodere (Encoders).
    b. Bytte til lavere frekvenser.
    c. Utnytte interferensen som oppstår når to mottakere står i nærheten.
    d. Øke båndbredden.

17. Når det gjelder adresse-feltene i gammel (versjon 4) og ny IP-versjon (versjon 6), så er det slik at
    a. De bruker like mange bit.
    b. Den gamle versjonen har ikke nok bit til å adressere alle maskinene på Internett.
    c. Ny versjon bruker dobbelt så mange bit.
    d. Ny versjon klarer seg med halvparten så mange bit.

18. Hva er programvareevolusjon?
    a. Programvare som må endres pga. av endringer i utføringsmiljøet.
    b. Kjøre programvare på raskere datamaskiner.
    c. Programvare som blir stadig større og raskere.
    d. Programvare som blir mer og mer intelligent og fleksibel.

1. Anta at vi har en todimensjonal liste (liste av lister) med navn. For eksempel `lister = [ ['Anh, Ine', 'By, Ken', …], ['By, Ken', 'Cox, Jo', …], …]` Her er hver indre liste alfabetisk sortert etter etternavn og uten duplikater, men samme navn kan forekomme i flere av de indre listene. Du kan også anta at det er ca. like mange navn i hver liste som det er lister i den store listen. Vi ønsker en funksjon `antall_n (lister, navn)` som returnerer hvor mange av listene det gitte navnet fins i. Her er pseudokode for en slik løsning, bestående av to funksjoner, hvor den ene – `antall_n ()` – kaller den andre som heter `antall ()`.

```
function antall_n (lister, navn):
      ant ← 0
      la liste_n gå fra og med første til og med siste element lister:
            ant ←ant + antall (liste_n, navn) # funksjonen fra pseudokoden
            # for antall() under, vil gi 0 eller 1
      returner ant

function antall (liste, navn):
      antall ← 0
      la n gå fra og med første til og med siste element i liste:
            hvis n == navn:
                  antall ←antall + 1
      returner antall
```

   Kompleksiteten til denne løsningen vil være:
   a. $\Theta(n)$
   b. $\Theta(n \log n)$
   c. $\Theta(n^2)$
   d. $\Theta(n^3)$

2. I stedet for å bruke funksjonen `antall ()` inni funksjonen `antall_n`, kan vi bruke binærsøk, dvs. vi bytter ut `ant ←ant + antall (liste_n, navn)` *i* pseudokoden med `ant ←ant + bin_search (liste_n, navn)` *hvor* det kan antas at `bin_search` i dette tilfellet returnerer 1 hvis `navn` fins i `liste_n` og 0 hvis det ikke fins. Kompleksiteten til `antall_n` vil da bli:
   a. $\Theta(n)$
   b. $\Theta(n \log n)$
   c. $\Theta(n^2)$
   d. $\Theta(n^3)$

Oppgave 2 Programmering binærkoding (25%)

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 man lage funksjoner for å lese en tekstfil med binærkode og gjøre om dette til tegn kodet i eget kodesett for tegn og bokstaver og lagre det til en tekstfil.

Oppgave 2a (5%)

Lag funksjonen `load_bin` som har en inn-parameter `filename`, som er navnet på fila som skal lastes inn. Funksjonen skal lese inn alt innholdet i fila og returnere innholdet som en tekststreng uten linjeskift eller mellomrom. Fila som det leses fra er en tekstfil bestående av binære tall (0 og 1). Hvis fila ikke eksisterer eller ikke kan åpnes, skal funksjonen returnere en tom streng samt skrive ut følgende feilmelding til skjerm: "Error: Could not open file <filename>", der <filename> er navnet på fila.

Eksempel på kjøring av fila "binary-file.txt" som inneholder følgende:

```
0100010
001
1001010101
11011
```

```
>>>
>>> load_bin("binary-file.txt")
'01000100011001010010111011'
>>>
```

Eksempel på kjøring av fila "binary-fill.txt" som ikke eksisterer:

```
>>>
>>> load_bin("binary-fill.txt")
Error: Could not open file binary-fill.txt
''
>>>
```

Oppgave 2b (5%)

Lag funksjonen `bin_to_dec` som har en inn-parameter `binary`, som er en tekststreng av ukjent størrelse bestående av binære tall (tekststreng med nuller og enere). Funksjonen skal returnere et heltall (dvs. i titallssystemet) som tilsvarer det binære tallet angitt med tekststrengen `binary`. Oppgaven ***skal ikke løses*** ved hjelp av innebygde funksjoner for å oversette binærtall til heltall.

Eksempel på bruk av funksjonen:

```
>>>
>>> bin_to_dec ("101")
5
>>> bin_to_dec ("11111111")
255
>>> bin_to_dec ("10000000000")
1024
>>>
```

Oppgave 2c (4%)
Lag funksjonen `dec_to_char` som har en inn-parameter `dec`, som er et heltall med verdi mellom 0 og 31. Funksjonen skal returnere et tegn eller en bokstav avhengig av verdien av `dec`:

- Hvis `dec` har verdien 0 skal tegnet for " " (mellomrom) returneres
- Hvis `dec` har verdien 1 skal tegnet for "," (komma) returneres
- Hvis `dec` har verdien 2 skal tegnet for "." (punktum) returneres
- Hvis `dec` har en verdi mellom 3 og 31 skal en stor bokstav i det norske alfabetet returneres, der `dec=3` gir bokstaven `"A"`, `dec=4` gir bokstaven `"B"`, helt opp til `dec=31` som gir bokstaven `"Å"`.
- For alle andre verdier av `dec` skal funksjonen returnere en tom streng.

Eksempel på bruk av funksjonen:

```
>>> dec_to_char (0)
' '
>>> dec_to_char (1)
','
>>> dec_to_char (2)
'.'
>>> dec_to_char (3)
'A'
>>> dec_to_char (31)
'Å'
>>>
```

Oppgave 2d (4%)
Lag funksjonen `bin_to_txt` som har en inn-parameter `binstring`, som er en tekststreng av ukjent lengde bestående av binære tall (tekststreng med nuller og enere). Funksjonen skal returnere en tekststreng bestående av bokstaver og tegn som er kodet i henhold til oppgave 2 c) der hvert tegn er representert med 5 bit. Inn-parameteren `binstring` vil alltid være et multiplum av fem siffer.

Eksempel på bruk av funksjonen:

```
>>>
>>> bin_to_txt ("00010")
'.'
>>> bin_to_txt ("00011")
'A'
>>> bin_to_txt ("101000010001101")
'RBK'
>>>
```

Oppgave 2e (7%)

Lag funksjonen `main`, uten parametere. Funksjonen skal gjøre følgende:

1. Skrive ut teksten "Binary-to-text converter" til skjerm
2. Spørre brukeren om navn på fil det skal lastes fra (tekstfil som inneholder binære tall) og ta vare på filnavnet i variabelen `b_file`.
3. Laste inn fila `b_file` og lagre innholdet av fila til variabelen `b_string` (tekststreng av bestående av binære tall)
4. Oversette tekststrengen av binære tall til tekst med bokstaver og tegn og lagre innholdet i variabelen `txt`.
5. Spørre brukeren om navn på navn på fil som resultatet skal lagres til og ta vare på filnavnet i variabelen `t_file`.
6. Skrive innholdet av variabelen `txt` til fila med filnavn angitt i variabelen `t_file`.
7. Skrive ut til brukeren at: «`<b_file>` has been converted and saved to `<t_file>`»

Hvis funksjonen får problemer med å skrive til fila, skal følgende feilmelding skrives:" Error: Could not write to file `<t_file>`". (Det som står mellom `< >` i disse utskriftene skal byttes ut med innholdet til variablene, jfr. eksempel under)

Eksempel på kjøring av funksjonen:

```
>>>
>>> main ()
Binary-to-text converter
Name of binary file to load from: binary.txt
Name of text file to save to: out.txt
binary.txt has been converted and saved to out.txt
>>>
```

Innhold binary.txt :

```
01100101111010110110
010111000000000
0010000101100111100100
001111010000000
0100001110100011000 1
1010000000
0010101110001110001 1
1000001011100000 1001
00010
```

Innhold out.txt :

```
JUSTIN BIEBER FLOOR CLEANING.
```

Oppgave 3 Programmering Allidrett (30%)

(I denne oppgaven kan det være gunstig å kalle funksjoner som du har laget i tidligere deloppgaver. Selv om du ikke har fått til den tidligere oppgaven, kan du kalle funksjon derfra med antagelse om at den virker som spesifisert i oppgaveteksten.)

Lea og Lars er trenere for Allidrett for 2.klassingene på Pythonmyra skole. De deltar i miniturneringer i fotball og innebandy, med 3-5 spillere på banen og gjerne 1-2 innbyttere i tillegg. Da det tar noe tid å utarbeide manuelle planer for laginndeling og innbyttefrekvens til disse kampene, ønsker de et program som kan gjøre noe av jobben.

Oppgave 3a (4%)
Hvis man har innbyttere på et lag, er det ønskelig at alle får like mye spilletid i hver kamp. Skriv en funksjon `sek_paa_benken(ant_paa_laget, ant_paa_banen, kamptid)`. Inn-parameterne er hhv. hvor mange spillere det er på laget, hvor mange av disse som skal være på banen til en hver tid, og varigheten av en kamp i antall minutter. Funksjonen skal returnere hvor mange sekunder hver enkelt spiller må være innbytter i løpet av en kamp, avrundet til nærmeste heltall. Eksempel på kjøring:

```
>>> sek_paa_benken(6, 5, 12)
120
>>> sek_paa_benken(6, 4, 12)
240
>>> sek_paa_benken(7, 5, 12)
206
>>>sek_paa_benken(5, 5, 12)
0
>>>
```

Oppgave 3b (4%)
Når trenerne ser på klokka under kamper for å sjekke når neste innbytte skal skje, er det enklere å forholde seg til minutter og sekunder enn kun sekunder. Skriv derfor en funksjon `minutt_sekund(sekunder)` hvor innparameter er antall `sekunder` (heltall) og returverdi en streng på formen `'mm:ss'`. Hvis antall minutter er mindre enn 10, skal strengen ha formen `'m:ss'`, mens sekunder alltid skal representeres som to tegn, også om det er mindre enn 10. Eksempel på kjøring

```
>>> minutt_sekund(120)
'2:00'
>>> minutt_sekund(206)
'3:26'
>>>
```

Oppgave 3c (4%)
Det hender ofte at enkelte må melde forfall til en cup. Skriv en funksjon `les_inn_forfall( )` som skal la brukeren skrive inn ett og ett navn via tastatur, eller helt tom streng (") for å avslutte (ingen flere forfall). Navnene som brukeren skriver inn, skal legges i en liste som skal returneres fra funksjonen. Du kan anta at brukeren ikke gjør noen tastefeil, så all input er korrekt.
Eksempel på kjøring (ledetekstene skrives ut av funksjonen, navnene skrives inn av bruker, og lista er det som funksjonen returnerer):

```
>>> les_inn_forfall()
Skriv navn, eller kun ENTER (tom tekst) for å avslutte.
Spiller som har meldt forfall: Henrik
Spiller som har meldt forfall: Emma B.
Spiller som har meldt forfall: Lucas
Spiller som har meldt forfall:
['Henrik', 'Emma B.', 'Lucas']
>>>
```

Oppgave 3d (4%)
Skriv en funksjon `finn_tilgjengelige (alle, forfall)` som tar inn som parametere to lister, hvor den første er alle barna som er med på allidretten, og den andre er de som har meldt forfall til en viss cup. Funksjonen skal returnere en ny liste som kun inneholder de barna som er tilgjengelige for å spille den aktuelle cupen. NB: Funksjonen må ikke endre innhold på den lista som gis inn som argument for parameteren `alle`.

Eksempel på kjøring:

```
>>> barn = ['Ada', 'Bo', 'Emma A.', 'Emma B.', 'Henrik',  'Ine', 'Jo', 'Kim',
'Lucas', 'My', 'Ola', 'Pia']
>>> forfall = ['Henrik', 'Emma B.', 'Lucas']
>>> finn_tilgjengelige(barn, forfall)
['Ada', 'Bo', 'Emma A.', 'Ine', 'Jo', 'Kim', 'My', 'Ola', 'Pia']
>>>
```

Oppgave 3e (6%)

Enkelte barn (og dels også foreldre) har sterke meninger om hvem som helst vil være på lag sammen, og dette kan være en kilde til konflikter. For å bli minst mulig påvirket av dette ønsker trenerne at programmet skal sette opp forslag til laginndeling automatisk, basert på en helt tilfeldig fordeling av spillere. Skriv en funksjon `laginndeling(spillere, sp_per_lag)` hvor innparameteren `spillere` er listen av de som er tilgjengelige for den aktuelle cupen, og `sp_per_lag` er antall spillere som man skal ha på banen. Funksjonen skal returnere en liste av lister, hvor den ytre lista inneholder de lagene man skal stille med i denne cupen, og de indre listene inneholder navn på spillerne på hvert av disse lagene. Alle lag må ha minst det antall spillere som innparameteren `sp_per_lag` angir, og ingen person kan være satt opp for å spille på mer enn ett lag. For øvrig skal funksjonen sette opp det maksimale antall lag man kan spille med (dvs. færrest mulig innbyttere), så det blir mest mulig spilletid på barna.

Eksempel på kjøring: 14 barn skal delta i en cup hvor hvert lag skal ha 4 på banen, funksjonen har tilfeldig fordelt barna i 3 lag, hvor 2 har 5 spillere (dvs. 1 innbytter) og ett har 4 spillere (ingen innbytter). Dette lagoppsettet returneres som en liste av lister.

```
>>> sp = ['Ada', 'Bo', 'Eli', 'Isa', 'Cindy', 'Henrik',  'Ine', 'Jo', 'Kim',
'Lucas', 'My', 'Noor',  'Ola', 'Pia']
>>> laginndeling(sp, 4)
[['Eli', 'Henrik', 'Pia', 'Ada', 'Ine'], ['Kim', 'My', 'Ola', 'Lucas', 'Noor'],
['Isa', 'Bo', 'Cindy', 'Jo']]
>>>
```

Oppgave 3f (4%)

Skriv en funksjon `main()` som leser inn fra brukeren hvilke barn som har meldt forfall, hvor mange spillere det skal være på banen (per lag) og hvor lang kamptid (minutter) det er i en gitt cup. Du kan anta at en global variabel `BARN` inneholder en liste over alle barna som er med på allidretten, så disse trenger ikke leses inn. På bakgrunn av de gitte opplysningene skal `main()` skrive ut opplysninger til skjerm som vist nedenfor (eksemplet viser kjøring under antagelse av at `BARN` inneholdt Jo og Henrik samt de som deretter er listet opp på de tre lagene under):

```
Skriv navn, eller kun ENTER (tom tekst) for å avslutte.
Spiller som har meldt forfall: Jo
Spiller som har meldt forfall: Henrik
Spiller som har meldt forfall:
Spillere per lag: 5
Kamptid (minutter): 15
Lag 1 :
['Pia', 'Bo', 'Ada', 'Lucas', 'Emma A.', 'Cindy']
Tid på benken per spiller: 2:30

Lag 2 :
['Emma B.', 'Yngve', 'Ola', 'My', 'Quentin', 'Sara']
Tid på benken per spiller: 2:30

Lag 3 :
['Noor', 'Kim', 'Tuva', 'Rashad', 'Ine']
Tid på benken per spiller: 0:00
```

Oppgave 3g (4%)

Hver minicup er lagt opp slik at hvert lag spiller et antall kamper, typisk 3-4. Kampoppsett for cupen sendes ut av arrangøren på en tekstfil med formatet hvor hver linje inneholder klokkeslett (for kampstart), de to lagene som skal spille mot hverandre, og hvilken bane de skal spille på. Trenerne ønsker at programmet skal ha en funksjon, `ny_fil()`, som leser fila med kampoppsettet (kampoppsett.txt), og skriver en ny tekstfil på samme format – ourGames.txt – som KUN inneholder de kampene som involverer lag fra Pythonmyra. La funksjonen ha tre inn-parametere: En for navnet til laget (f.eks. Pythonmyra), en for navnet på fila som skal leses inn (f.eks. kampoppsett.txt) og en for navnet på fila som det skal skrives til (f.eks. ourGames.txt).

Eksempel på utdrag fra fila kampoppsett.txt:

```
17:20 Pythonmyra 1 - Ranheim 2 (Bane 3)
17:20 Freidig 1 - Ranheim 3 (Bane 1)
17:20 Ranheim 1 - Utleira 2 (Bane 2)
17:20 Astor 1 - Vestbyen (Bane 4)
17:40 Freidig 4 - Pythonmyra 2 (Bane 1)
17:40 Trond 1 – Freidig 2 (Bane 2)
17:40 Pythonmyra 3 – Trond 2 (Bane 3)
18:00 Utleira 2 – Pythonmyra 1 (Bane 2)
```

Med fila gitt over skulle funksjonen resultere i at det lages en ny fil som følger:

```
17:20 Pythonmyra 1 - Ranheim 2 (Bane 3)
17:40 Freidig 4 - Pythonmyra 2 (Bane 1)
17:40 Pythonmyra 3 – Trond 2 (Bane 3)
18:00 Utleira 2 – Pythonmyra 1 (Bane 2)
```

Fila kan i det generelle tilfelle selvsagt være lenger enn det som er vist her. Lag funksjonen som produserer fila som kun inneholder Pythonmyras kamper.

Oppgave 4 Kodeforståelse (20%)

Oppgave 4a (5%)

Hva blir skrevet ut om en kaller `a(5)` med koden vist under? (3 %)
Forklar med en setning hva funksjonen `a()` gjør? (2 %)

```
def a(b):
    for c in range (1, b+1):
        for d in range (1, b+1):
            if (d<b):
                print (c*d, end=',')
            else:
                print(c*d)
```

Oppgave 4b (5%)

Hva returneres hvis man kaller `f([[3,5],[2,4],[1,3]])` med koden nedenfor? (3 %)
Forklar med en setning hva funksjonen `f()` gjør? (2 %)

```
def f(b):
    c=len(b[0])
    d=len(b)
    g = [[0 for row in range(d)]
         for col in range(c)]
      for e in range (0, c):
        for f in range (0, d):
            g[e][f] =b[f][e]
    return g
```

Oppgave 4c (5%)

Hva returneres hvis man kaller `u(5)` med koden nedenfor? (3 %)
Forklar med en setning hva funksjonen `u()` gjør? (2 %)

```
def u(x):
  if x <=1:
    return 1
  else:
    return x*u(x-1)
```

Oppgave 4d (5%)

Hva returneres hvis man kaller `nrk('Nylnpuokrtrjhtrsklkiok'` med koden nedenfor? (3 %)

Forklar med en setning hva funksjonen `nrk()` gjør? (2 %)

```
def nrk(tekst):
  s=''
  i=0
  j=1
  while i<len(tekst):
    s+=tekst[i]
    i=i+j
    j=j+1
  return s
```

Appendix: Some useful functions

FIX Round towards zero.

FIX(X) rounds the elements of X to the nearest integers towards zero.

FLOOR Round towards minus infinity.

FLOOR(X) rounds the elements of X to the nearest integers towards minus infinity.

FCLOSE Close file.

ST = FCLOSE(FID) closes the file associated with file identifier FID, which is an integer value obtained from an earlier call to FOPEN. FCLOSE returns 0 if successful or -1 if not.

FEOF Test for end-of-file.

ST = FEOF(FID) returns 1 if the end-of-file indicator for the file with file identifier FID has been set, and 0 otherwise.

The end-of-file indicator is set when a read operation on the file associated with the FID attempts to read past the end of the file.

FGETL Read line from file, discard newline character.

TLINE = FGETL(FID) returns the next line of a file associated with file identifier FID as a MATLAB string. The line terminator is NOT included. Use FGETS to get the next line with the line terminator INCLUDED. If just an end-of-file is encountered, -1 is returned.

FIND Returns the linear indexes of non-zero elements in a matrix.

FIND([0 1 0 1 0]) returns [2 4]. If the first parameter has more than one row, a column vector containing the linear indexes of non-zero elements are returned. An optional second parameter set the maximum number of indexes to return.

FOPEN Open file.

FID = FOPEN(FILENAME,PERMISSION) opens the file FILENAME in the mode specified by PERMISSION:

'r'	open file for reading
'w'	open file for writing; discard existing contents
'a'	open or create file for writing; append data to end of file
'r+'	open (do not create) file for reading and writing
'w+'	open or create file for reading and writing; discard existing contents
'a+'	open or create file for reading and writing; append data to end of file

FPRINTF Write formatted data to file.

COUNT = FPRINTF(FID,FORMAT,A,...) formats the data in the real part of array A (and in any additional array arguments), under control of the specified FORMAT string, and writes it to the file associated with file identifier FID. COUNT is the number of bytes successfully written. FID is an integer file identifier obtained from FOPEN. It can also be 1 for standard output (the screen) or 2 for standard error. If FID is omitted, output goes to the screen.

FORMAT is a string containing ordinary characters and/or C language conversion specifications. Conversion specifications involve the character %, optional flags, optional width and precision fields, optional subtype specifier, and conversion characters d, i, o, u, x, X, f, e, E, g, G, c, and s.

The special formats \n,\r,\t,\b,\f can be used to produce linefeed, carriage return, tab, backspace, and formfeed characters respectively. Use \\ to produce a backslash character and %% to produce the percent character.

INPUT Read a value from the keyboard and into a variable
ANSWER=INPUT(STR) prints STR as a prompt, reads a number and assigns it to ANSWER. If character string are to be read, use the optional second parameter ‘s’.

ISEMPTY - Determine whether array is empty
This MATLAB function returns logical 1 (true) if A is an empty array and logical 0 (false) otherwise.
TF = isempty(A)

LENGTH The length of vector.
LENGTH(X) returns the length of vector X. It is equivalent to MAX(SIZE(X)) for non-empty arrays and 0 for empty ones.

MAX finds the highest element in a vector, or the highest element in each column of a matrix.

MIN finds the lowest element in a vector, or the lowest element in each column of a matrix.

MOD Modulus after division.
MOD(x,y) is x - n.*y where n = floor(x./y) if y ~= 0.

RANDI Pseudorandom integers from a uniform discrete distribution.
R = RANDI(IMAX,N) returns an N-by-N matrix containing pseudorandom integer values drawn from the discrete uniform distribution on 1:IMAX.
RANDI(IMAX,M,N) or RANDI(IMAX,[M,N]) returns an M-by-N matrix.

REM Remainder after division.
REM(x,y) is x - n.*y where n = fix(x./y) if y ~= 0.

SIZE The size of array.
D = SIZE(X), for M-by-N matrix X, returns the two-element row vector
D = [M,N] containing the number of rows and columns in the matrix.

SQRT Square root.
SQRT(X) is the square root of the elements of X.

SSCANF Extracts values from a string according to a format string. Opposite of FPRINTF.
A=SSCANF(’12/11-2014’,’%d/%d-%d’) returns a column vector containing the values
12, 11, and 2014.

STRSPLIT Splits the first (string) parameter into a cell array of substrings, according to the delimiter string given as the second parameter. STRSPLIT('one, two, three', ', ') results in {'one', 'two', 'three'}. Multiple alternative delimiters can be specified using a cell array as the second parameter.

STRTOK separates the first token of a string from the rest of that string.
[TOKEN, REST]=STRTOK(‘ first second’, DELIM) sets TOKEN to ‘first’ and REST to ‘ second’.
The optional parameter DELIM contains a list of delimiter characters – where the space character is default. Any delimiter characters before the first token are ignored.

STR2NUM Convert string matrix to numeric array.
X = STR2NUM(S) converts a character array representation of a matrix of numbers to a numeric matrix.
For example, S=['12'; '34'] str2num(S) => [12; 34]

SUM The sum of elements.
S = SUM(X) is the sum of the elements of the vector X. If X is a matrix, S is a row vector with the sum over each column.

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				