INF 2820 V2016: Innleveringsoppgave 1

- **Besvarelsene skal leveres i devilry innen fredag 10.2 kl 18.00**
- **Det blir 5 sett med innleveringsoppgaver. Hvert sett gir inntil 20 poeng. Til sammen kan en få inntil 100 poeng. For å kunne gå opp til eksamen må du ha oppnådd minst 60 poeng til sammen på de 5 settene.**
- **For å oppnå poeng er det viktigere at du prøver å løse en oppgave enn at løsningen er helt riktig. Vektleggingen er slik sett annerledes enn på eksamen.**
- **Innleveringsfrister utsettes kun i forbindelse med sykdom (ved egenmelding eller legeerklæring), og omlevering vil ikke være mulig**
- **Vi går ut i fra at du har lest og er kjent med IFIs reglementet for obligatoriske oppgaver: http://www.mn.uio.no/ifi/studier/admin/obliger/index.html og også kjenner rutinene for behandling av fuskesaker http://www.uio.no/om/regelverk/studier/studier-eksamener/fuskesaker/**
- **Dette er en individuell oppgave. Det skal ikke leveres felles besvarelser.**
- **Du kan levere om igjen i Devilry inntil fristen, men inkluder i så fall for alt du vil i den siste innleveringen.**

Oppgave 1: Formelle språk (2 poeng)

a) La L = {ab, abbb, abc, c} og M = {bba, a}. Hva blir LM?
b) Hva blir ML?
c) Betrakt språket M*L. Hvilke av følgende uttrykk er i dette språket: *a, aa, ab, ac, acc, aacc, aaabbb*, ε?
d) Hvilke av disse uttrykkene er i språket (ML)*
e) La N=Ø og P={ε}. Hva blir LN og hva blir LP?

Innlevering: Svar på spørsmålene

Oppgave 2: Endelige tilstandsmaskiner (4 poeng)

Denne oppgaven kan gjøres i JFLAP. Du anbefales likevel å løse den med papir og penn først for å få eksamenstrening. Så kan du bruke JFLAP til å kontrollere løsningen din.

a. Lag en ikke-deterministisk endelig tilstandsmaskin (NFA) som beskriver språket L1=L(*a*b(a+c)* + ac(b+a)*), der alfabetet er A={*a, b, c*}. (Symbolet + er her disjunksjon).
b. Lag en deterministisk maskin (DFA) som beskriver det samme språket.
c. Lag en tilstandsmaskin som beskriver komplementspråket til L1.
d. Hvilke av følgende uttrykk er i L1?
    i. abc
    ii. acb
    iii. bac
    iv. bbc
    v. aaaa
    vi. aaab
    vii. aaba
    viii. abaa
    ix. abab
    x. baaa

Innlevering: Svar på de fire punktene. Hvis du løser oppgaven med papir og penn og tegner diagrammer, er det tilstrekkelig (for pkt a-c) å ta et tydelig bilde av hvert diagram og levere disse. Bruker du JFLAP, kan du lagre diagrammene som JPEG-filer og levere.

Oppgave 3: Regulære uttrykk (4 poeng)

Vi skal her bruke hva vi vil kalle "rene" regulære uttrykk. Det er de vi gikk gjennom på forelesningen bygget opp ved

Regulære uttrykk	Beskriver språket
Ø	L(Ø) = Ø
ε	L(ε) = { ε }
a, for alle a ∈ A	L(a) = { a }
Hvis R og S er regulære uttrykk:	
(R + S)	L(R+S)=L(R) ∪L(S)
(R T)	L(R T) = L(R)L(T)
(R*)	L(R*)=L(R)*

La alfabetet A={a, b, c}. Lag regulære uttrykk for følgende språk:

a) Ord som inneholder minst tre b-er på rad.
b) Ord som ikke inneholder mer enn to b-er på rad.
c) Ord hvor antall b-er er delelig med 3 eller antall a-er er delelig med 2 (eller begge deler).

Innlevering: De regulære uttrykkene.

Arbeid med tekst og med regulære uttrykk i Python

Vi går ut i fra at alle nå har kjennskap til Python og NLTK som beskrevet i notatet "Python og NLTK i INF2820, V2017". Vi skal her arbeide mer med tekster og regulære uttrykk i Python og NLTK. Vi vil buke kapittel 3 i NLTK-boka, http://www.nltk.org/book/.

Hvis du ikke er kjent med Python og Pythons strengebegrep bør du først av alt arbeide deg gjennom seksjon 3.2, "Strings: Text processing at the lowest level" og gjøre oppgavene fra seksjon 3.12: 2, 4, 5, 10, 11 (Ikke innlevering).

Vi kan bruke Pythons regulære uttrykk for å sjekke resultatene våre fra første del. Se seksjon 3.4 i NLTK-boka. Regulære uttrykk i unix-funksjoner som *grep* og i programmeringsspråk som Python avviker litt fra våre "rene" regulære uttrykk på to områder.

a) De prøver å finne ut om det regulære uttrykket matcher en del av en tekst. Vi er interessert i om det regulære uttrykket matcher en hel tekst (et helt ord.)
b) De har en rikere syntaks og tillater en del mer.

Til pkt. (b): Vi vil i første omgang begrense oss til det som svarer til de rene regulære uttrykkene, men skrive (a|b) i stedet for (a+b).

Til pkt. (a): For å kunne bruke Pythons re-modul til å sjekke om en hel streng er beskrevet av et regulært uttrykk eller ikke, kan vi skrive en liten kodebit.

```
>>> import re
>>> def beskriver(regeksp, ord):
      reg = "^("+regeksp+")$"
      if re.search(reg, ord):
            return True
      else:
            return False
```

Vi kan så sjekke om ulike strenger matcher det regulære uttrykket:

```
>>> regeks = 'b(a|b|c)*'
>>> beskriver(regeks, 'baa')
True
>>> beskriver(regeks, 'abc')
False
>>>
```

Oppgave 4 Regulære uttrykk i Python (2 poeng):

Test løsningene dine fra oppgave 3 i Python. Dvs. for hver av de tre oppgavene,

1. Skriv det regulære uttrykket som et Python-regulært uttrykk!
2. Lag to eksempler på strenger som skal være i språket og to som ikkje skal være det!
3. Test uttrykket ditt for de fire strengene!

Innlevering: For hvert punkt: uttrykket i Python, de fire strengene du vil teste på, resultatet av testing.

Oppgave 5: Innlesning av tekst (4 poeng)

Vi skal nå bevege oss fra de mer teoretiske eksemplene til språkteknologiske anvendelser. I dette emnet skal vi arbeide mye med tekster. Teksten "Python og NLTK i INF2820, V2017" er konvertert til ren tekst og lagret på IFIs maskiner her /projects/nlp/inf2820/Python_INF2820_v2017.txt.

a) Les den inn i en interaktiv Python-sesjon som en streng. Kall den "pyt_raw". Oppskriften finner du i seksjon 3.1 i NLTK-boka, underavsnitt "Reading local files". Hvis du ser noe rusk i teksten, så rens den.

b) Når vi videre skal arbeide med en tekst, kan det være en fordel å dele den opp i en liste av ord, der hvert ord er en streng. Den enkleste måten å gjøre dette på er ved å bruke split i python.
>>> pyt_words1 = pyt_raw.split()
NLTK gir oss også et annet alternativ:
>>> pyt_words2 = nltk.word_tokenize(pyt_raw)
Hva blir forskjellen på de to? Ser du fordeler ved å bruke word_tokenize? (Obs! NLTKs word_tokenize er optimalisert for engelsk og kan gi noen rare resultat for norsk.)

c) Et ord som "jeg" er det samme om det står først i en setning og skrives "Jeg". Tilsvarende for andre ord. For en del anvendelser er det derfor en fordel å gjøre om teksten til bare små bokstaver før vi går videre. Gjør om alle ordene i pyt_words2 til små bokstaver, og kall resultatet pyt_low.

d) Plukk ut alle ordforekomstene i pyt_low som inneholder en av de norske bokstavene æ, ø, å. Hvor mange slike ordforekomster er det?

e) Vi er nå interessert i hvor mange forskjellige ord det er i teksten som inneholder en av de norske bokstavene. Plukk ut de unike forekomstene. Hvor mange er det?

f) Vi skal nå skrive de unike forekomstene til en fil. Når vi skal skrive noe til en fil, kan vi åpne fila ved

```
>>> f=open(<filnavn>, 'w')
```

Vi kan skrive til fila ved å bruke

```
>>> f=write(<det vi vil skrive>)
```

Og til slutt lukke fila ved

```
>>> f.close()
```

For mer om dette se
https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

Skriv de unike forekomstene av ord som inneholder æ, ø eller å til en fil kalt norske.txt, ett ord på hver linje. Det kan være nyttig å vite at '\n' betyr ny linje i Python-strenger, prøv

```
>>> print('jeg\ner\n\nglad')
```

Innlevering: Svar på spørsmålene i (b), (d) og (e), forklar med kode hvordan du løste oppgaven. Levér også fila norske.txt.

Oppgave 6: Regulære uttrykk og bearbeiding av tekst (4 poeng)

I teksten vi har lest inn er det en del uttrykk som ikke er vanlige ord, inkludert

- stinavn på Unix-format, som "/projects/nlp/nltk_data"
- tall

For en del prosessering kan det være bedre å skifte ut slike uttrykk, som ofte bare forekommer en gang, med en betegnelse som sier oss at her står det et stinavn, eller her står det et tall.

a) Skriv et regulært uttrykk i Python for stinavn.

b) Skriv en prosedyre som bruker Pythons regulære uttrykk (seksjon 3.4 og delvis 3.5 i NLTK-boka) som tar en tokenisert tekst og skifter alle tokens som er stinavn med "<path>" og alle tokens som er tall med "<num>". Med en tokenisert tekst, mener vi her en liste av strenger som f.eks. pyt_words2). Bruk resultatet ditt på pyt_words2.

c) Vi vil i stedet skifte ut stinavn og tall på en tekst som ikke er tokenisert, dvs. en tekst som er en streng. Skriv en prosedyre som leser en streng og som skifter ut alle substrenger som er stinavn med "path" og alle substrenger som er tall med "<num>". Hint: Det enkleste her er å bruke sub-metoden i Pythons re-modul, se: https://docs.python.org/2/howto/regex.html#search-and-replace

Innlevering: Det regulære uttrykket og prosedyrene det spørres etter.

SLUTT