UNIVERSITETET I AGDER

Gruppe5

Alle ICA

Terje Seljåsen, Mathias Hartveit, Stina Berg Welde, Johanne Mosling, Erik Haaland, Eirik Slagnes.

ICA 1

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke%204%20-%20ICA%201
Sist endret: 01.06.2016

1.2.1
Lise: log2(8/4) = 1 bit
Per: log2(8/5) = 0.67 = 1 bit
Oskar: log2(8/3) = 1.41 = 1 bit
Louise: 3 bit

1.2.2
a)
Fordi en 1bits kode tar mindre plass og X har størst sannsynlighet, derfor vil den forekomme oftere. Derfor er det smartere og gi den 1 bit.
b) 00101001100000 = X X Y Y X Z X X X X X
c) (python)

1.2.3
a)
b)
c) Den er veldig effektiv. Fordi den bruker dict. Som base og henter alt ut fra den. Den kan derfor brukes på alt i det det engelske språket.

1.2.4
(?)

ICA 2

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke%205%20-%20ICA%202
Sist endret: 03.06.2016

Table 1

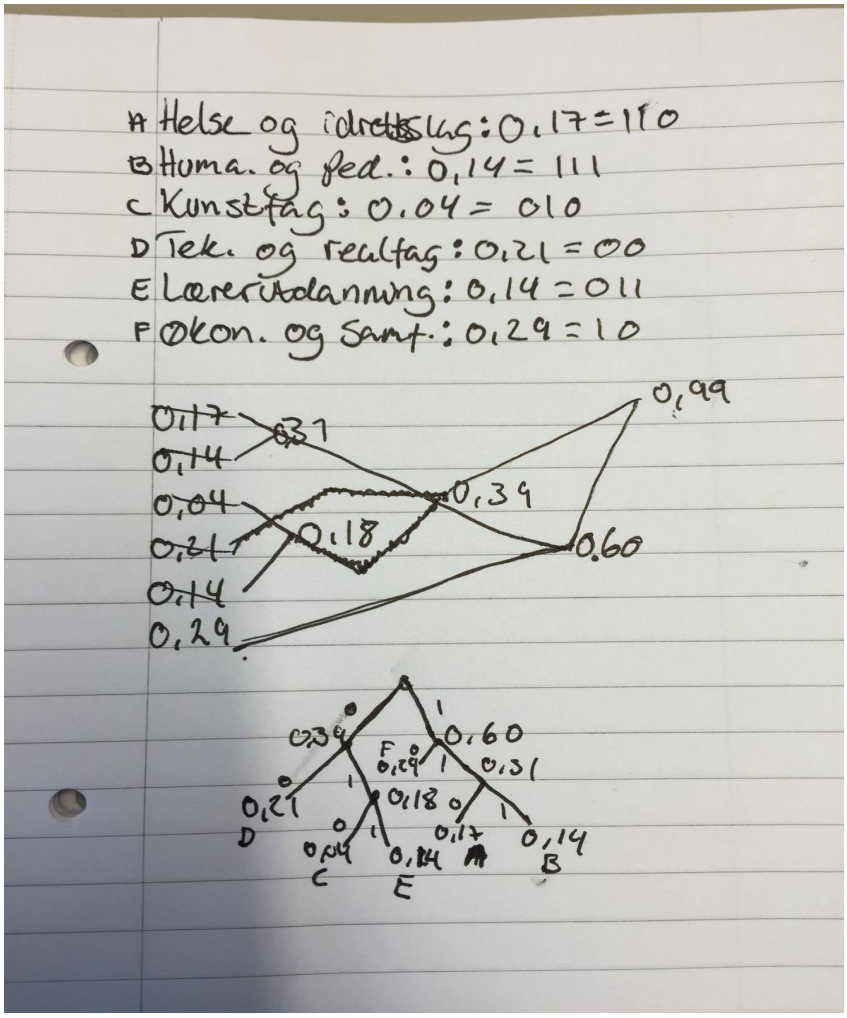
Fakultet	Ant. Studenter	Sannsynlighet
Helse- og Idrettsfag	1829	17%
Humaniora og Pedagogikk	1525	14%
Kunstfag	420	4%
Teknologi og Realfag	2166	21%
Lærerutdanning	1506	15%
Økonomi og Samfunnsvitenskap	3093	29%
Total	**10539**	**100%**

Basert på illustrasjon over, lag en tabell med kolonner "UiAs fakultet", "Antall studenter" og "Sannsynlighet" (for at en UiA student hører til det spesifikke fakultetet) for høstsemester 2014

Når du lærer om at en tilfeldig valgt student hører til et spesifikt fakultet, for hvilket fakultet får du MINST informasjon?

Minst informasjon får man ved Økonomi og Samfunnsfag

Design en variabel-størrelse Huffman kode, som minimaliserer gjennomsnittsantall av bits i en melding som er kodet for fakulteter og tilfeldig valgte grupper av studenter. Vis et binært tre for Huffman kode og gi kode og kodelengde for hver kurs.



Hva er gjennomsnittslengden for en melding som inneholder fakultetskoder for 100 tilfeldig valgte studenter?

- Øko: 29x2 = 58
- Lærer: 14x3 = 182
- Tek: 21x2= 42
- Kunst: 4x3= 12
- Hum: 14x3= 42
- Helse: 17x3= 51

  = 387bits

Hvordan forholder denne lengden seg til entropien av sannsynlighets distribusjon i dette tilfelle? Beregn og forklar.

- Økonomi og Samfunnsvitenskap: 1,78
- Teknologi og Realfag: 2,25
- Helse- og Idrettsfag: 2,55
- Humaniora og Pedagogikk: 2,83
- Lærerutdanning: 2,83
- Kunstfag: 4,64
- Gjennomsnittslengde = 2,81
- Det fakultetet som forekommer oftest er det som inneholder mest informasjon, og får en større komprimeringsgrad

Lag en algoritme som kan kode og dekode med den valgte koden. Algoritme skal først forklares og så implementeres i Python. (github)

ICA 3

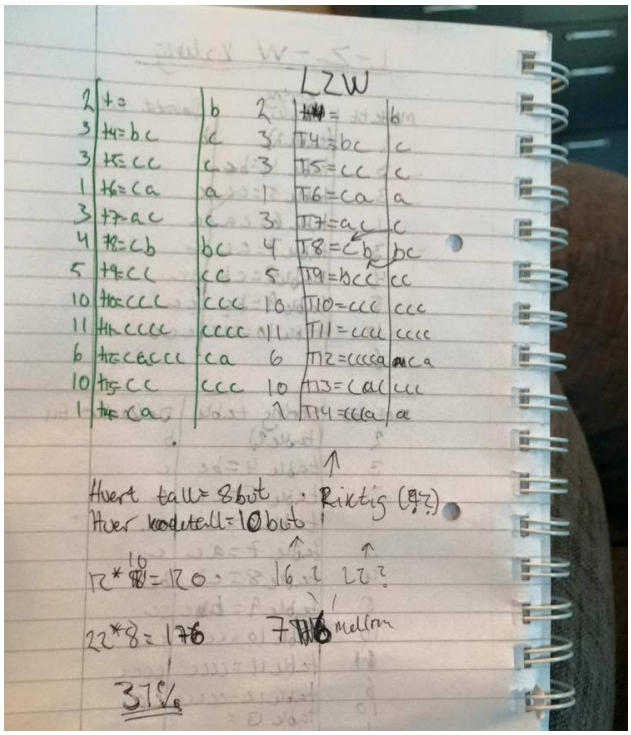
b) Implementer LZW algoritme i Python basert på pseudokode fra teksten (teori for uke05).

Vi valgte å importere en LZW algoritme fra nettet for å spare tid.

https://pypi.python.org/pypi/lzw/
https://github.com/joeatwork/python-lzw

c) Beregne komprimeringsgraden (i prosent) for dette tilfelle:

Våre beregninger tilsier at komprimeringsgraden for LZW i dette tilfellet er 31%



d) Basert på frekvensen av forekomsten for de tre symbolene(a, b, c), lag en Huffman kode, kode meldingen og beregn prosentvis komprimeringsgrad. Hvordan er den i forhold til LZW caset? Forklar.

(Se bilder nedenfor!)

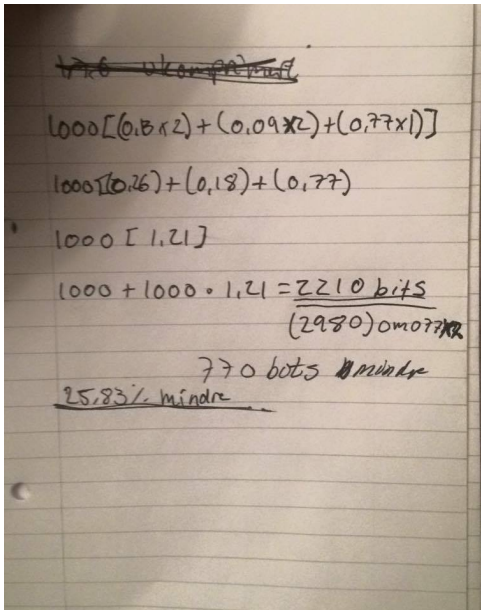
Huffman koden har en komprimeringsgrad på 25.83%. Dette er betydelig mindre enn komprimeringsgrad ved bruk av LZW-metoden. I meldingens nåværende form vil antall bytes spart være få, men ved større filer eller meldinger vil forskjellene bli større.

f) Finne komprimeringsgraden for begge tekstfilene. Reflekter!

Vi kodet og komprimerte to tekstfiler: Hamlet på 184KB og Complete Shakespeare på 5.6MB(5590KB). Etter koding var størrelsen på Hamlet 98KB og Complete Shakespeare 2.9MB(2863KB)

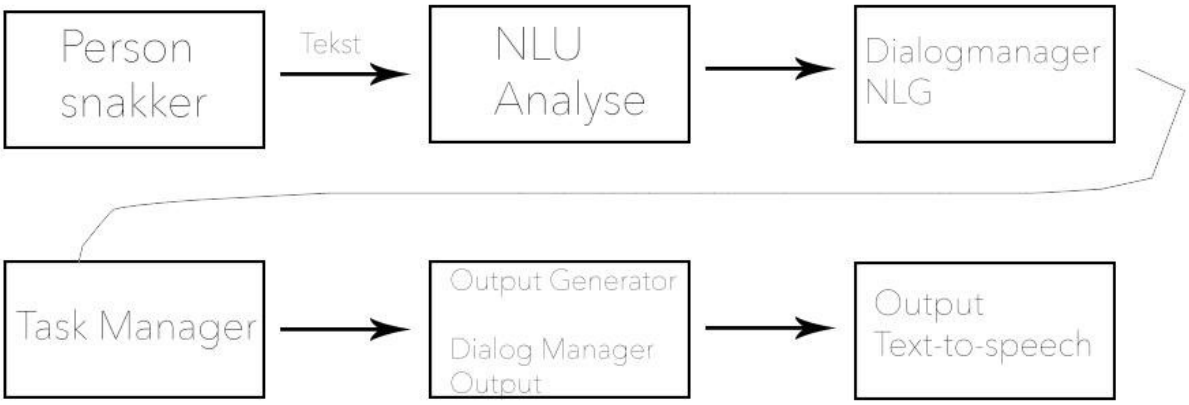
Det vil si at komprimeringsgraden for Hamlet var på ca 53%, mens for Complete Shakespeare var den ca 51%. Prosentvis er ikke forskjellen stor men jo større filen er jo større lagringsplass spares ved komprimering. (50% av 1000 er mer en 50% av 100.)

LZW	Hoff
2 = B	10
3 = C	0
3 = C	0
1 = A	11
3 = C	0
4 = BC	100
5 = CC	00
10 = CCC	000
11 = CCCC	0000
6 = CA	011
10 = CCC	000
1 = A	11
31 %	25,83%



ICA 4
Dialog System

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke%207%20-%20ICA%204
Sist endret: 26.05.2016



1. Person snakker (Lyd)
Lyden blir konvertert til tekst med en input recognizer/ decoder
i dette tilfelle tar vi utgangspunkt i en akustisk modell for dekoding av brukerens input (lyd). En akustisk modell er en fil so inneholder statistiske representasjoner av hver distinkte lyd som utgjør et ord. I det engelske språket er det 40 forskjellige "lyder" som brukes i talegjenkjenningen. hvis vi for eksempel skal analysere ordet "house" vil lydene vi leter etter være "hh, aw, s". Det som skjer i taledekoderen er at hvis den finner disse lydene så vil den gjøre en antakelse om at ordet som er sagt er "house". Når dekoderen finner en av de 40 distinkte lydene noterer den det ned, og slik fortsetter den til brukeren er ferdig å snakke. Når brukeren er ferdig sjekker dekoderen lydene den fant opp mot lyder som utgjør ord i "ordboken".

2. NLU
Teksten blir analysert av NLU (Natural Language Understanding engin) som registrer navn, ordklassen til ordene som blir brukt, og sammensetningen av hele setninger for å registrer syntaks og sammenhenger. Det kan ofte bli en utfordring siden det oppstår ukjente og uventede innspill i inputet, og systemet må derifra funne ut hva som er riktig syntaks.

3. Dialogmanager NLG
Informasjons- semantikken fra NLU blir analysert av dialogmanageren som også holder historikk og status i dialogen. Dialogmanagaeren er ansvarlig for statusen og flyten i samtalen. Den viktigste rollen til DMen er å bestemme hvilken handling dialog agenten(?) skal ta til en hver tid. En måte for DMen å tolke den menneskelige

inputen på er å lete etter stikkord i det som blir sagt, og generere et svar utifra dette. Outputen til dialogmanageren er instrukser til de andre komponentene i dialogsystemet. Dette blir konvertert til menneskelig språk av en NLG (Natural Language Generation) komponent. outputen blir gjort mer naturlig ved å huske dialog historikken.

4. Task manager
Dialog manageren tar som regel kontakt med en eller flere task managere som har kunnskap om det spesifikke domenet.

5. Output generator
Før dialogmanageren kan gi en output, og før outputen kan gis til brukeren, må den få hjelp av en output generator. Det er flere typer output generatorer, bla. Natural language generator. NLG sin oppgave er å generere naturlig språk fra maskinens representasjonssystem til en mer logisk form. Altså å gå fra maskinens ”språk” til et naturlig et. Alternativer er Layout engine og Gesture generator.

6. Output
Til slutt vil outputet bli gitt tilbake til brukeren ved hjelp av en output renderer i form av tekst eller lyd. Systemet kan bruke en "text-to-speech engine" for å gi output i form av lyd, og mange systemer bruker animasjoner av mennesker eller dyr når de gir outputet tilbake til brukeren. Eks. "Eviebot".

Kilder:
http://www.voxforge.org/home/docs/faq/faq/what-is-an-acoustic-model
https://en.wikipedia.org/wiki/Dialog_system
https://en.wikipedia.org/wiki/Part_of_speech
https://en.wikipedia.org/wiki/Natural_language_understanding
https://en.wikipedia.org/wiki/Dialog_manager
https://en.wikipedia.org/wiki/Chatterbot

ICA 5

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke-8-ICA-5
Sist endret: 03.06.2016

Kartlegge alle datamaskiner som gruppemedlemmene bruker og finne så mange detaljer om deres bestanddeler / komponenter som mulig (CPU frekvens, størrelse for RAM, L1, L2, osv.). Gi også en kort beskrivelse av bestanddelene (hva er deres funksjon og hvordan de fungerer) samt lag en tabell og presenter deres funn (ta også med detaljer om operativsystem).

CPU frekvens - *Hvor fort prosessoren jobber.*
L2 & L3 - *Minne i CPU. Veldig lavt minne, men utrolig raskt.*
RAM - *Har begrenset minne. Lagrer informasjon midlertidig så det raskt kan hentes opp.*
OS - *Operativsystem*

	Terje Mac-book pro	**Erik** Lenovo	**Eirik** Leno-vo	**Mathias** Mac-book Air	**Johanne** Mac-book Air	**Stina** Macbook Pro)
CPU frekvens	IntelCore i5 2,7GHz Dual Core	A10-7300 with Radeon™ R6 Graphics 1.90GHz Quad core	IntelCore i7 4720HQ - 2,7GHz Quad Core	Intel Core i7 1,7 GHz Dual Core	Intel Core i5 1,6 GHz Dual Core	Intel Core i5 2,3 GHz Dual Core
RAM	8GB	8GB	16GB	8GB	4GB	4GB
L2	256KB (?)	4MB	6MB	256 KB	256KB	256KB
L3	3MB (?)			4 MB	3MB	3MB
OS	OS X 10.11.3 (15D21)	Windows 10	Windows 10	OS X 10.11.3(15D21)	OS X 10.11.3 (15D21)	OS X 10.11.3 (15D21)

Søk for å finne tall 150128 av 1000000.

Ytelse	**Slow Search**	**Fast Search**
Terje	0.623	0.088
Stina	0,756	0,265
Johanne	0,780	0,573
Erik	2.666	0.402

Ytelse	Slow Search	Fast Search
Eirik	0.546	0.079
Mathias	0.651	0.111

Søk etter ordet "denmark" i Hamlet.txt fra uke 06

Ytelse	Slow Search	Fast Search
Terje	0.021	0.016
Erik	0.082	0.065
Eirik	0.017	0.018
Stina	0,079	0,054
Johanne	0,029	0.016
Mathias	0.028	0.014

Algoritmen:

```
def search_fast(haystack, needle):
    for item in haystack:
        if item == needle:
            return True
    return False
```

Algoritmen vi har brukt er O(N) algoritme innenfor BIG 'O algoritmene. O(N) algoritmen er lineær, noe som tilsier at en vekst i datagruppen (i dette tilfellet haystack) vil lede til en lik proporsjonal økning i søketiden, ettersom tiden det tar å søke gjennom hvert element er den samme. Eneste forskjellen mellom seach slow og search fast er at search slow må gå gjennom hele listen før den returnere svar.
Hvis needle er lengre fremme i listen vil algoritmen returene fortere.

Link til Mathplotlib python prosjekt basert på tabellene ovenfor: https://github.com/Math-ih13/IS-105_2016_Gruppe-5/releases/tag/uke5

ICA 6

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke-9-ICA-6
Sist endret: 03.06.2016

a)
(definisjons-, begrepsnivå): Se tabbelen for NAND "gate" og lag tilsvarende tabeller for NOT, AND, OR, XOR og NOR.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

A	B	OR
0	0	0
0	1	0
1	0	1
1	1	1

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

A	NOT
0	0
0	0
1	1
1	1

b) Hvilken operasjon gjør programmet i tabellen under på en Turing maskin utgangspunkt i illustrasjonen rett under spørsmålet? **Legger til verdien "1" til tallet "7".**
Hva står på "tape" når prosessen når "Stop Tilstand"? **100**
Hvor mange operasjoner kreves det for å komme til "Stop Tilstand"? **9stk**
Forklar resultatet.

tid-spunkt	Tilstand	Oppdager,	Skriver	Flytter til	Ny tilstand
t0	0	blank	blank	høyre	1
t1	1	1	0	høyre	1
t2	1	1	0	høyre	1
t3	1	1	0	høyre	1
t4	1	blank	1	venstre	2
t5	2	0	0	venstre	2
t6	2	0	0	venstre	2
t7	2	0	0	venstre	2
t8	2	blank	blank	høyre	halt

c)

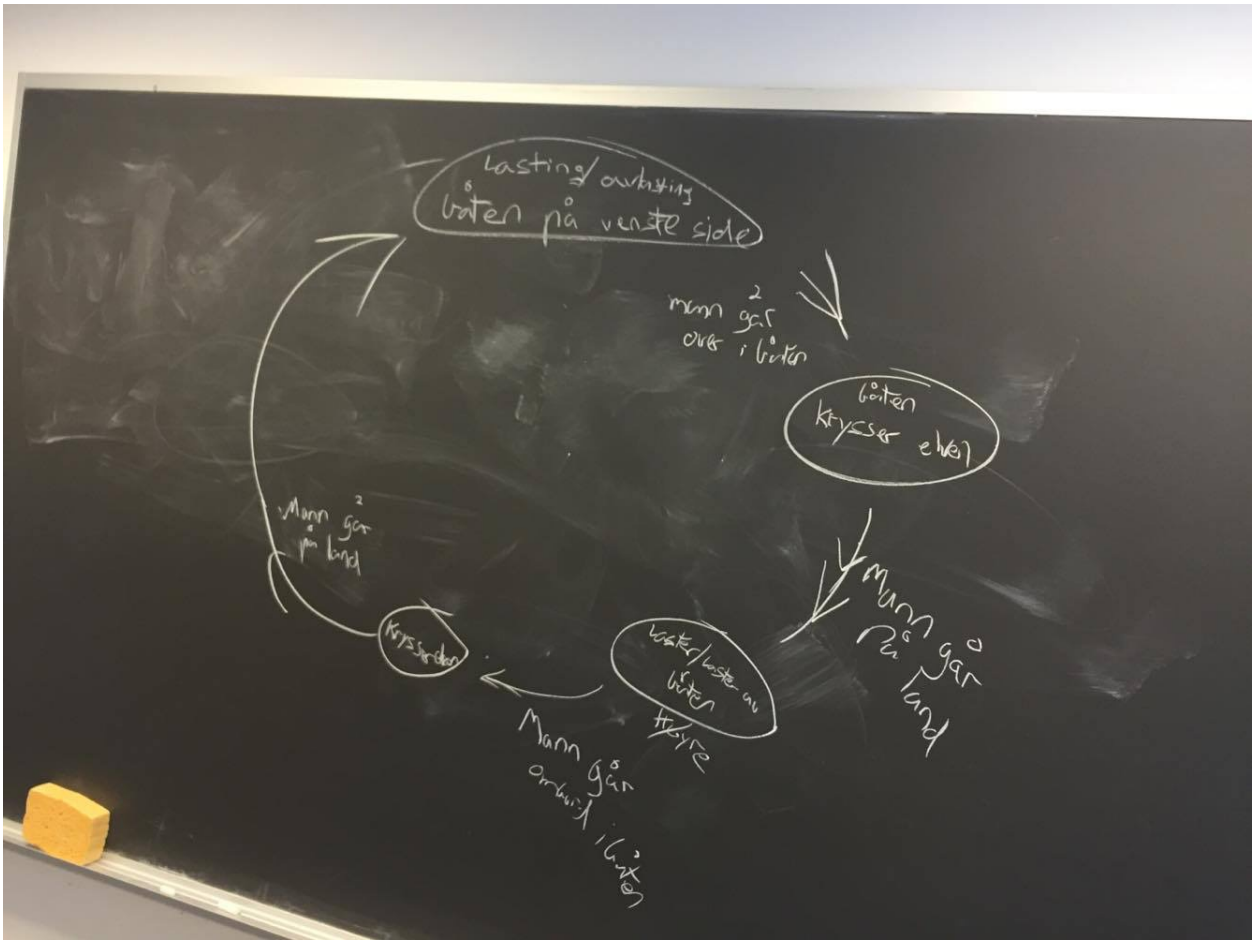
In the parking gate controller task, the machine has four possible states.

'Waiting' - for car to arrive at the gate
'Raising' - for raising the arm
'Raised' - the arm is at the top position and we're waiting for the car to drive through the gate
'Lowering' - lowering the arm after the care has passed through the gate

Båten på venstre side. (laster på/av)
    Man går i båt
Båten krysser
    Man går på lang
Båten på høyre side (laster på av)
    Man går i båt
Båten krysser

Vi kan sette opp en formel for å se alle mulige tilstander verdenen kan ha ved å se på hvor reven, kornet og kyllingen kan være. Det er 3 objekter som kan være på 4 forskjellige plasser.
81
Det er 81 mulige tilstander verdenen kan ha.

d) https://github.com/Mathih13/IS-105_2016_Gruppe-5/commit/a9d3bd71c0265772-ca46c92ef13a8cf39aa08fa7



ICA 7

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke-10-ICA-7
Sist endret: 01.06.2016

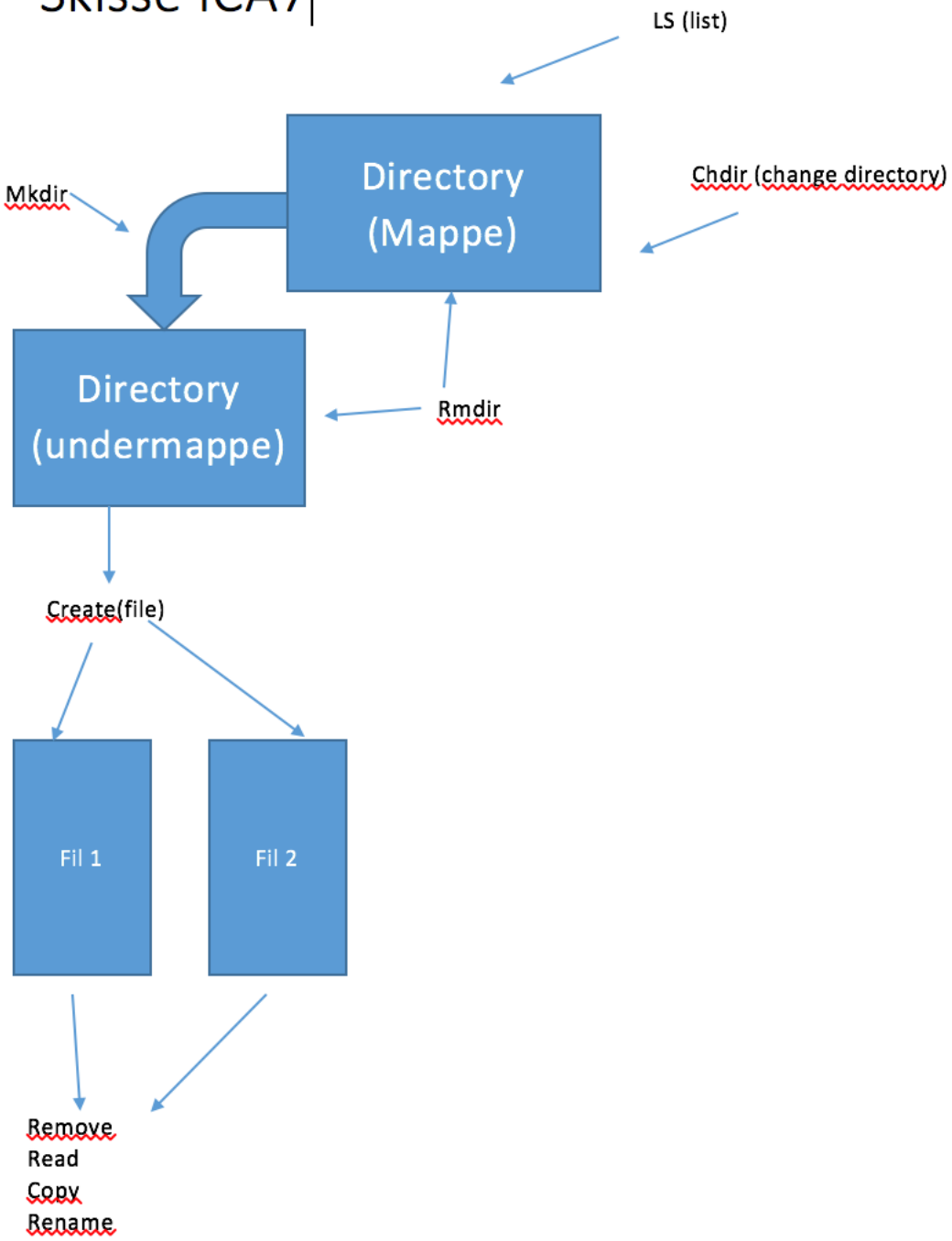
Målet med dette prosjektet er å implementere ett enkelt filsystem på toppen av en virtuell disk. For å oppnå dette, vil du opprette et bibliotek som tilbyr et sett av grunnleggende fil systemkall (for eksempel åpne, lese, skrive, ...) til programmer. Fildata og filsystem meta-informasjonen vil bli lagret på en virtuell disk som er en enkel fil som lagres i standardplasseringen (CWD - Current Working Directory), som er den gjeldende arbeidskatalogen. Den virtuelle disken blir da som sagt lagret som en enkel fil på det "ekte" filsystemet levert av operativsystemet du bruker. Det vil si at du er i utgangspunktet implementerer filsystemet på toppen av operativsystemets filsystem.
Systemet lagrer alle filene i en hoved-directory på den virtuelle disken, derfor trenger ikke systemet å støtte et katalog-hierarki.

Funksjonene:

mkdir	- lage mappe
chdir	- bytte til en annen mappe
rmdir	- slette mappe
rm	- slette fil
ls	- skrive ut innhold i aktuell mappe
create	- lage ny fil
read	- lese og skrive ut innholdet i en fil
mv	- flytte fil/gi nytt navn
cp	- lage en kopi av filen
help	- liste av funksjoner
exit	- avslutt programmet

Funksjonene er nærmere beskrevet i koden.

Skisse ICA7


LS (list)
Directory (Mappe)
Mkdir
Chdir (change directory)
Directory (undermappe)
Rmdir
Create(file)
Fil 1
Fil 2
Remove
Read
Copy
Rename

ICA 8

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke-14-ICA-8
Sist endret: 03.06.2016

I denne oppgaven har vi brukt Tkinter for den grafiske presentasjonen til River prosjektet.

ICA 9

https://github.com/Mathih13/IS-105_2016_Gruppe-5/tree/master/Uke-15-ICA-9
Sist endret: 03.06.2016

ICA 9 inneholder samling av ICA 8 samt restrukturering og støtte for kommunikasjon over nettverk via python sockets modulen. Foreløpig vil bruk av “get in” og “get out” gi svar i form av oppdatert database(model) fra serveren. Det visuelle oppdateres ikke foreløpig pga tidsmessige begrensninger.