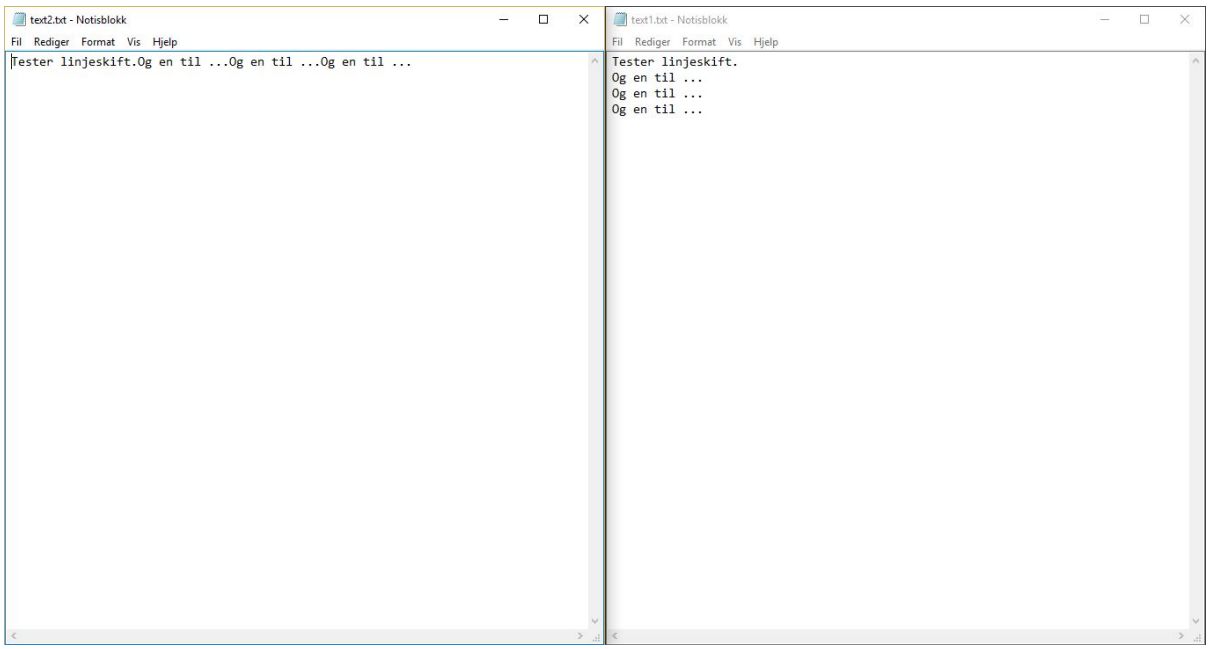
Ica 4
Navnliste:
Brede Knutsen Meli
Nils Fredrik
Eirik Aanestad Fintland
Jan Kevin Henriksen
Mats Skjærvik
Mikael Kimerud
Abdikani Gureye
Morten Johan Mygland

1:

a)



(Bildet til venstre)
54 65 73 74 65 72 20 6C 69 6E 6A 65 73 6B 69 66 74 2E 0A 4F 67 20 65 6E 20 74 69 6C 20 2E 2E 2E 0A 4F 67 20 65 6E 20 74 69 6C 20 2E 2E 2E 0A 4F 67 20 65 6E 20 74 69 6C 20 2E 2E 2E 0A
[T e s t e r l i n j e s k i f t .
O g e n t i l . . .
O g e n t i l . . .
O g e n t i l . . .
(Bildet til høyre)
54 65 73 74 65 72 20 6C 69 6E 6A 65 73 6B 69 66 74 2E 0D 0A 4F 67 20 65 6E 20 74 69 6C 20 2E 2E 2E 0D 0A 4F 67 20 65 6E 20 74 69 6C 20 2E 2E 2E 0D 0A 4F 67 20 65 6E 20 74 69 6C 20 2E 2E 2E 0D 0A
[T e s t r l i n j e s k i f t .
O g e n t i l . . .
O g e n t i l . . .
O g e n t i l . . .

Som vi kan se i text1.txt er byten 0D lagt til 4 ganger, noe som forklarer hvorfor den ene filen er 4 bytes større enn den andre.. 0D er sammen med 0A kommandoen for linjeskift.

b)

Linjeskift m/ punktum, mac: Vi ser tegnet 0A fordi tekstfiler opprettet i unix og nyere mac versjoner kun bruker line feed, mens de i de første mac versjonene kun brukte carriage return og tegnet 0D

```
]bredes-mbp:fileutils bredemeli$ go run fileutils_main.go
0A 2E 0A 2E 0A 2E 0A 2E 0A 2E 0A
```

linjeskift windows: Dette er samme fil som i illustrasjonen over. Siden filen er opprettet i windows, så vil tegnene 0D og 0A dukke opp når en trykker på enter tasten. 0A står for line feed, mens 0D står for carriage return. Vi har brukt en funksjon for å telle antall "\x0A" i filen, som vi får ut som tekst.

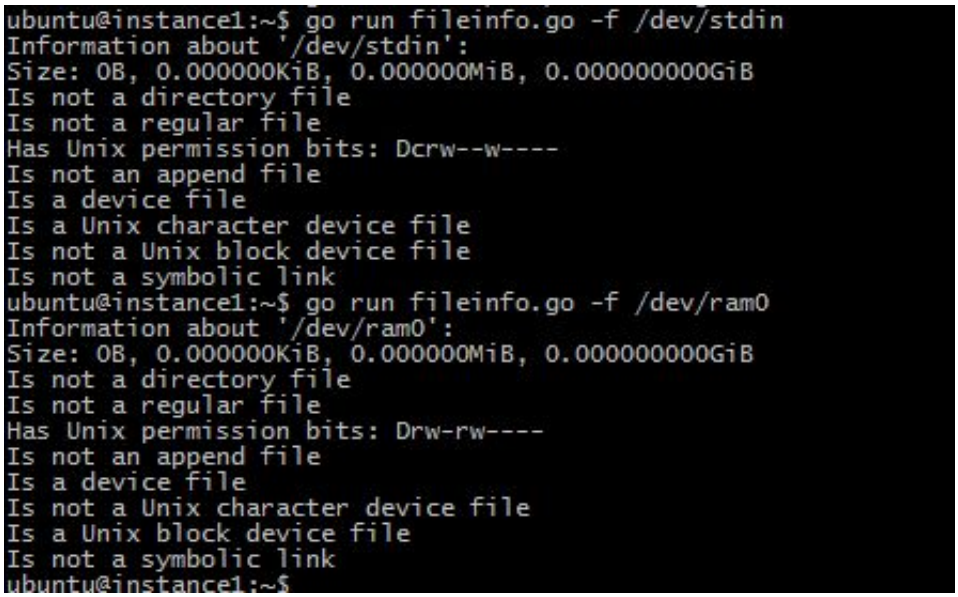
```
Nils Fredrik@Nils MINGW64 ~/documents/is105/ica04/is105-ica04/files/fileutils (nils)
$ go run main.go
 E 0D 0A 2E 0D 0A 2E 0D 0A 2E 0D 0A [.
 .
 .
 .
]
Det er 4 linjeskift.
```

2:

a)

```
ubuntu@bredeinstans:~$ go run fileinfo.go treasure.txt
Information about '/home/ubuntu/treasure.txt':
Size: 2089B, 2.040039KiB, 0.001992MiB, 0.000001946GiB
        Is not a directory
        Is a regular file
Unix permission bits: -rw-r--r--
        Is not append only
        Is not a device file
        Is not a Unix character device
        Is not a Unix block device
        Is not a symbolic link
```

b)



/dev/stdin

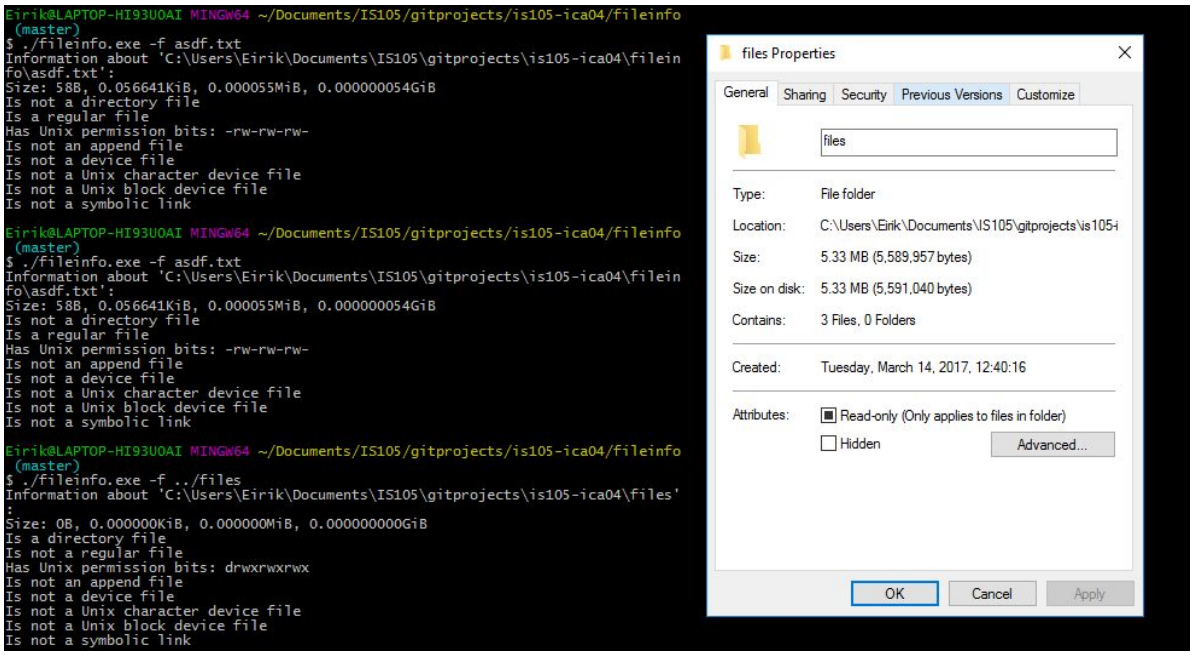
Dette er en device file/unix device file som har Unix permissions bits: Dcrw--w---- som tilsvarer Owner: read write, Group: write, Others: ingen tilgang. D står for “door”, c står for “device character”.

/dev/ram0

Dette er en devicefile men ikke en Unix character device file. Filen har en Unix permission bits: Drw-rw---- som tilsvarer Owner: read write, Group: read write, Others: ingen tilgang. D står for “door”.

c)

Ut i fra dette skjermbildet kan vi se at selve mappen ikke har en størrelse i følge fileinfo.exe, selv om mappen inneholder 5.33MB av filer. Dette vil sannsynligvis si at win10 ikke teller selve mappen som en fil, men heller innholdet av mappen. Vi testet dette videre ved å lage en tom mappe på win10 og OSX, og kunne se at den tomme mappen vises som 0 bytes på win10, men 68 bytes på OSX. Dette er fordi OSX bygger på unix-prinsippet at alt er en fil, mens Windows ikke gjør dette.



3:

a)

Det finnes en rekke ulike metoder for å jobbe med filer i GO programmer. Ved hjelp av pakken “os” kan vi lese filer og redigere filer. Det nødvendig å sjekke for feil, når det er gjort så kan man minske en fils innhold i minnet. Det er vanlig å ville styre hvordan og hvilke deler av en fil som skal leses, og da kan man enten lese stream av bytes med buffer eller uten buffer. Man kan også bruke Seek funksjon for å finne frem til et spesifikt punkt i filen og lese derfra, og vi kan velge å lese et gitt antall bytes.

Vi kan bruke “Bufio” pakken til å lage en buffer skriver som gjør at vi kan jobbe med en buffer i minnet før vi skriver det over til disk. dette er praktisk hvis man ikke ønsker å bruke mye tid på diskens IO. Det er også nyttig hvis man vil skrive en byte om gangen og deretter lagre det i minne før man dumper alt til filen på èn gang, fremfor å skrive IO for hver byte. Vanlig buffer størrelse er på 4096 bytes, og minimum er på 16 bytes.

b)

Vi har testet hvilke karakterer som forekommer flest ganger i filen “pg100.txt”, samt antall linjeskift.

```
Det er 39878 'T' i denne filen
Det er 14169 'U' i denne filen
Det er 3587 'V' i denne filen
Det er 16508 'W' i denne filen
Det er 608 'X' i denne filen
Det er 9128 'Y' i denne filen
Det er 532 'Z' i denne filen
Det er 245509 'a' i denne filen
Det er 46768 'b' i denne filen
Det er 67194 'c' i denne filen
Det er 134216 'd' i denne filen
Det er 406157 'e' i denne filen
Det er 69103 'f' i denne filen
Det er 57328 'g' i denne filen
Det er 218875 'h' i denne filen
Det er 199130 'i' i denne filen
Det er 2788 'j' i denne filen
Det er 29345 'k' i denne filen
Det er 146532 'l' i denne filen
Det er 95890 'm' i denne filen
Det er 216805 'n' i denne filen
Det er 282560 'o' i denne filen
Det er 46849 'p' i denne filen
Det er 2414 'q' i denne filen
Det er 209907 'r' i denne filen
Det er 215605 's' i denne filen
Det er 291243 't' i denne filen
Det er 115225 'u' i denne filen
Det er 34077 'v' i denne filen
Det er 73155 'w' i denne filen
Det er 4681 'x' i denne filen
Det er 85549 'y' i denne filen
Det er 1098 'z' i denne filen
Det er 78216 '.' i denne filen
Det er 83315 ',' i denne filen
Det er 8840 '!' i denne filen
Det er 10476 '?' i denne filen


Det er flest av 'e' i denne filen og det er 406157 stk
Det er nest flest av 't' i denne filen og det er 291243 stk
Det er tredje flest av 'u' i denne filen og det er 115225 stk
Det er fjerde flest av 'y' i denne filen og det er 85549 stk
Det er femte flest av ',' i denne filen og det er 83315 stk

Det er 124787 linjeskift i denne filen.
```

c)

```
Jan Kevin@DESKTOP-S6LQCVA MINGW64 ~/Documents/UIA Studie dokumenter/2.Semester/IS 105/Offline arbeid/ICA4/Oppgave
(master)
$ go test -bench=.
BenchmarkFinnLiten-4                2000            634404 ns/op
BenchmarkFinnMiddels-4              1000           2164422 ns/op
BenchmarkFinnStor-4                   10         166910160 ns/op
BenchmarkFinnBufferLiten-4         10000            205131 ns/op
BenchmarkFinnBufferMiddels-4        5000            205533 ns/op
BenchmarkFinnBufferStor-4          10000            203057 ns/op
BenchmarkFinnNBytesLiten-4         10000            202458 ns/op
BenchmarkFinnNBytesMiddels-4        5000            210737 ns/op
BenchmarkFinnNBytesStor-4          10000            200630 ns/op
PASS
ok      _/C_/Users/Jan_Kevin/Documents/UIA_Studie_dokumenter/2.Semester/IS_105/Offline_arbeid/ICA4/Oppgave_3/oppg
38s
```

4:

a)

Antall fakulteter: 6

Antall studenter(2014):

Helse og idrett:	1829
Humaniora og pedagogikk:	1525
Kunstfag:	420
Teknologi og realfag:	2166
Lærerutdanning:	1506
Økonomi og samfunnsvitenskap:	3093
totalt =	10´539

Tallene under er rundet av til nærmeste hele prosent

Helse og idrett:	1829 = 17%
Humaniora og pedagogikk:	1525 = 14%
Kunstfag:	420 = 4%
Teknologi og realfag:	2166 = 21%
Lærerutdanning:	1506 = 14%
Økonomi og samfunnsvitenskap:	3093 = 30%
total prosent:	100%

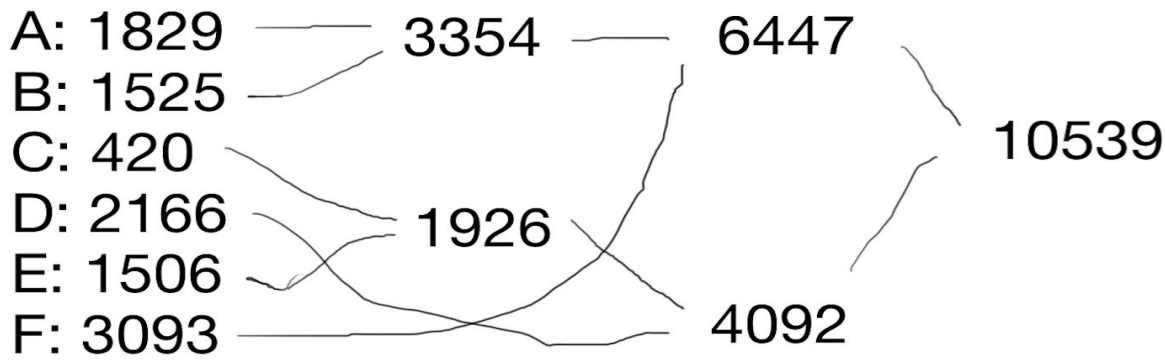
På bakgrunn av tallene ovenfor kan man for eksempel komme frem til at det er 17% sannsynlighet at en tilfeldig student tilhører Helse og idrett.

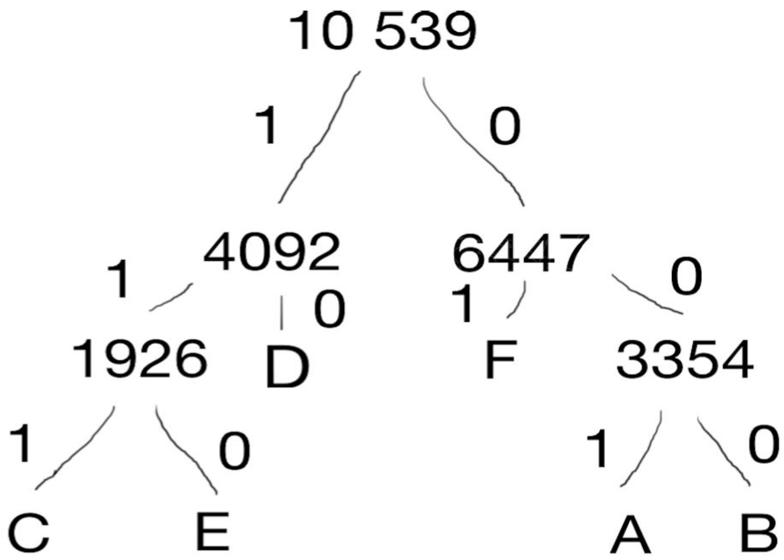
b)

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

Kunstfaget, er det fakultetet man får minst informasjon fra (4%). Eller får du minst informasjon fra det største fakultetet, altså økonomi??

c) Binært tre for huffmankode nedenfor med kodelengde for hvert fakultet.





Helse og idrett	A:001
Humaniora og pedagogikk	B:000
Kunstfag	C:111
Teknologi og realfag	D:10
Lærerutdanning	E:110
Økonomi og samfunnsvitenskap	F:01

d)

Hva er gjennomsnittslengden for en melding som inneholder fakultets koder for 100 tilfeldig valgte studenter? Beregn og forklar.

Hvis vi følger sannsynligheten, blir antallet studenter, eller bokstaver:
17A, 14B, 4C, 21D, 14E, 30F (fant ut hvor stor antall % fra hvert fakultet tidligere).
AKA: 17x”001”, 14x”000”, 4x”111”, 21x”10”, 14x”110”, 31x”01”(Antallet*bits, så A=17*3 bits)
AKA: 51+42+12+42+42+62
AKA: 251 (bits/bytes/tall???)

<u>Koden ser slik ut:</u>

0010010010010010010010010010010010010010010010010010000000000000000000000
0000000000000000000011111111111110101010101010101010101010101010101010101
0110110110110110110110110110110110110110110110101010101010101010101010101
01010101010101010101010101010101

e)

Leverer dessverre uferdig på deloppgave e, ettersom vi ikke greide å lage en algoritme som kan kode og dekode huffman koden. Vi fikk satt opp en struct, og laget en funksjon som inneholdt nøkkel, men kom ikke lenger.