UIA IS-105 UTKAST ICA01
Sist oppdater: 24. Januar 2017

ICA01

Besvarelsen skal evalueres med karakter (A-F). Begrunn alle svar!

1.1 Formål

Konfigurere diverse verktøy og bli kjent med shell/kommandolinje/terminal. Forstå grunnlaget for informasjonsteori.

For å konfigurere github, kan du lese GitHub dokumentasjon https://help.github.com/
For å kunne samarbeide vha. Github, kan du lese http://nvie.com/posts/a-successful-git-branching-model/ (husk at det er viktig å avtale en "flow-modell" som passer for deres team og oppgavene dere skal løse)
For å bli kjent med Golang, kan du ta en "Tour Of Go" https://tour.golang.org/welcome/1 (i denne ICA-en forventes at du tar "welcome" og "basics" avsnittene).
For å bli bedre kjent med Go, bør du installere utviklingsmiljøet på din datamaskin https://golang.org/doc/install

Aktuelt
Forståelse av teori, - infomasjonsteori.
Generell kunnskap
It i praksis, tallsystemer, koder.
Spesifikk kunnskap
Bruke Git sammen. Bli kjent med Golang gjennom praktiske ekspempler. Øke forståelsen av informasjonsteori.
Praktisk anvendelse
Logikk- og programmeringsoppgaver.

1.2 Oppgaver

1.2.1 Binære tall

Konverter følgende desimaltall til 2-tallssystemet (binært tallsystem):

(1) 1
(2) 2
(3) 5
(4) 8
(5) 16
(6) 256

For hvert svar oppgi også et antall bits som kreves for å representere det desimale tallet binært. Dere må også beskrive metoden dere brukte for å gjøre konverteringen.

Konverter følgende binære tall til desimaltall (mest signifikante bit-en er til venstre):

(1) 100
(2) 1001
(3) 1100110011

Beskriv kort metoden for konverteringen.

1.2.2 Informasjonsmengde

Flere personer prøver å gjette et tresifret (3-bit) binært tall.

(1) Lise har fått vite / lærer at tallet er et oddetall.
(2) Per har fått vite at tallet er IKKE et multiplum av 3 (dvs. ikke 0, 3, 6).
(3) Oskar har fått vite at tallet inneholder nøyaktig 2 enere.
(4) Louise har fått vite alt det Lise, Per og Oskar vet.

Hvor mye informasjon (i bits) har hver spiller fått?

1.2.3 Arbeid med git

Opprett en git repository “is105-uke04” på Github med README, .gitignore (for Golang) og lisensfil.

Konfigurer din maskin og klone repository på din maskin. Gjør alt i kommandolinje i ditt operativsystem! Noter alle kommandoer.

Bruk **`hello.go`** fra eksemplet https://tour.golang.org/welcome/1 og legg det inn i mappestrukturen til den klonede repository "is105-uke04". Legg filen i "staging" og gjør en "commit" med påfølgende "push".

1.2.4 Samarbeid i Git og Introduksjon i Golang

---

1) Velge en repository som hovedrepository og gi alle gruppemedlemmene tilgang til denne repository-en. Denne repositorien blir også sted for innlevering av all kode. All kode dere skriver videre i denne oppgaven skal lagres i denne repositorien.
   Hvilken fordeler og ulemper har en git-flow-modell med en hovedrepository?

2) Hver av gruppemedlemmene legger inn en endring (valgfritt) i filen **`hello.go`** i "master" branch.

   Installer også go på deres datamaskin https://golang.org/doc/install
   Programmer i go utføres med kommandoet **`go run <filnavn>`**
   Da genereres det ingen objektfil (en fil som man kan utføre på en platform[1], og hvis format er platformavhenging). Hvis man ønsker å generere en objektfil, er kommandoet **`go build <filnavn>`**
   Finn ut hva heter objektfiler for de mest brukte platformer (Unix/Linux, MS Windows, Mac OS X)! Hvorfor, etter deres mening, har disse platformene så forskjellige objektfil-formater?

3) Gå gjennom https://tour.golang.org/basics/1 Hvilke forskjeller ser dere i forhold til programmeringsspråket Java? (forutsetter at alle er kjent med Java, hvis ikke, så kan man også sammenligne med andre programmeringsspråk)

4) Implementer en pakke log, som inneholder en funksjon som beregner logaritme av base 2 (søk i Golang dokumentasjonen for å finne aktuelle funksjoner). Lag en mappe **`log`** som inneholder to filer, - **`log.go`** og **`main.go`**. I **`log.go`** implementerer dere selve beregningen av logaritmen (som en

[1] Det finnes ingen presis definisjon av begrepet platform, men det brukes ofte for å illustrere forskjeller på kombinasjonen mellom maskinvare og operativsystem. De fleste avhengigheter, som gjør at et program generert (kompilert / bygget og linket) på et system, kan ikke utføres på et annet system, kan relateres til forskjeller i platformen, dvs. en kombinasjon mellom maskinvaren og operativsystemet.

funksjon), mens i **`main.go`** bruker dere funksjonen fra **`log.go`**, og skriver ut resultatet til kommandolinje. Hvilke viktige poeng illustrerer denne øvelsen når det gjelder bruk av et programmeringsmiljø på en platform?

5) Bruk eksemplet fra https://gobyexample.com/command-line-arguments og lag et utførbart program **`logcli`**, som kan ta et argument (tallet som man skal beregne logaritmen av base 2 for); Er det hensiktsmessig å legge inn denne filen i git repository? Begrunn svaret!

6) Implementer en ny funksjon i filen **log**, som kan beregne både logaritmen med base 2 og logaritmen med base 10. Denne funksjonen skal ta to argumenter, base og tall, og returnere et beregnet resultat for logaritme med base for tall. Lag en utførbart program **logbcli**, som tar to argumenter og skriver ut resultatet til kommandolinje:

```
./<filnavn_programmet> 2 3
1.5849625007211563
```

Hvordan skiller pakken **`log`**, som dere har implementer, seg fra andre pakker i go, som, for eksempel, **`fmt`**?

1.3 Referanser

V. Driessen. (2010). A successful Git branching model. Retrieved from ULR http://nvie.com/posts/a-successful-git-branching-model/

GitHub Help, Sist sett 2017-01-24 på https://help.github.com/

P. Krill. (2017). Go tops Java, C, Python for programming language of the year. Retrieved from URL http://www.infoworld.com/article/3155924/application-development/go-tops-java-c-python-for-programming-language-of-the-year.html?imm_mid=0ec6d0&cmp=em-prog-na-na-newsltr_20170121

SLUTT.

JG/2017-01