Learning to Program with F#
Exercises
Department of Computer Science
University of Copenhagen

Jon Sporring, Martin Elsman, Torben Mogensen, Christina Lioma

August 24, 2020

0.1 Mastermind

0.1.1: I skal programmere spillet Mastermind. Minimumskrav til jeres aflevering er:

- Det skal være muligt at spille bruger mod bruger, program mod bruger i valgfrie roller og program mod sig selv.
- Programmet skal kommunikere med brugeren på engelsk.
- Programmet skal bruge følgende typer:

```
type codeColor =
  Red | Green | Yellow | Purple | White | Black
type code = codeColor list
type answer = int * int
type board = (code * answer) list
type player = Human | Computer
```

  hvor `codeColor` er farven på en opgavestift; `code` er en opgave bestående af 4 opgavestifter; `answer` er en tuple hvis første element er antallet af hvide og andet antallet af sorte stifter; og `board` er en tabel af sammenhørende gæt og svar.
- Programmet skal indeholde følgende funktioner:

```
    makeCode : player -> code
```

  som tage en spillertype og returnerer en opgave enten ved at få input fra brugeren eller ved at beregne en opgave.

```
    guess : player -> board -> code
```

  som tager en spillertype, et bræt bestående af et spils tidligere gæt og svar og returnerer et nyt gæt enten ved input fra brugeren eller ved at programmet beregner et gæt.

```
    validate : code -> code -> answer
```

  som tager den skjulte opgave og et gæt og returnerer antallet af hvid og sort svarstifter.
- Programmet skal kunne spilles i tekst-mode dvs. uden en grafisk brugergrænseflade.
- Programmet skal dokumenteres efter fsharp kodestandarden
- Programmet skal afprøves
- Opgaveløsningen skal dokumenteres som en rapport skrevet i LaTeX på maksimalt 20 oversatte sider ekslusiv bilag, der som minimum indeholder
  - En forside med en titel, dato for afleveringen og jeres navne
  - En forord som kort beskriver omstændighederne ved opgaven
  - En analyse af problemet
  - En beskrivelse af de valg, der er foretaget inklusiv en kort gennemgang af alternativerne
  - En beskrivelse af det overordnede design, f.eks. som pseudokode
  - En programbeskrivelse
  - En brugervejledning

  - En beskrivelse af afprøvningens opbygning
  - En konklusion
  - Afprøvningsresultatet som bilag
  - Programtekst som bilag

Gode råd til opgave er:

- Det er ikke noget krav til programmeringsparadigme, dvs. det står jer frit for om I bruger funktional eller imperativ programmeringsparadigme, og I må også gerne blande. Men overvej i hvert tilfælde hvorfor I vælger det ene fremfor det andet.
- For programmet som opgaveløser er det nyttigt at tænke over følgende: Der er ikke noget "intelligens"-krav, så start med at lave en opgaveløser, som trækker et tilfældigt gæt. Hvis I har mod på og tid til en mere advanceret strategi, så kan I overveje, at det totale antal farvekombinationer er $6^4 = 1296$, og hvert afgivne svar begrænser de tilbageværende muligheder.
- Summen af det hvide og sorte svarstifter kan beregnes ved at sammenlinge histogrammerne for de farvede stifter i hhv. opgaven og gættet: F.eks. hvis opgaven består af 2 røde og gættet har 1 rød, så er antallet af svarstifter for den røde farve 1. Ligeledes, hvis opaven består af 1 rød, og gættet består af 2 røde, så er antallet af svarstifter for den røde farve 1. Altså er antallet af svarstifter for en farve lig minimum af antallet af farven for opgaven og gættet for den givne farve, og antallet af svarstifter for et gæt er summen af minima over alle farver.
- Det er godt først at lave et programdesign på papir inden I implementerer en løsning. F.eks. kan papirløsningen bruges til i grove træk at lave en håndkøring af designet. Man behøver ikke at have programmeret noget for at afprøve designet for et antal konkrete situationer, såsom "hvad vil programmet gøre når brugeren er opgaveløser, opgaven er (rød, sort, grøn, gul) og brugeren indtaster (grøn, sort, hvid, hvid)?"
- Det er ofte sundt at programmere i cirkler, altså at man starter med at implementere en skald, hvor alle de væsentlige dele er tilstede, og programmet kan oversættes (kompileres) og køres, men uden at alle delelementer er færdigudviklet. Derefter går man tilbage og tilføjer bedre implementationer af delelementer som legoklodser.
- Det er nyttigt at skrive på rapporten under hele forløbet i modsætning til kun at skrive på den sidste dag.
- I skal overveje detaljeringsgraden i jeres rapport, da I ikke vil have plads til alle detaljer, og I er derfor nødt til at fokusere på de vigtige pointer.
- Husk at rapporten er et produkt i sig selv: hvis vi ikke kan læse og forstå jeres rapport, så er det svært at vurdere dens indhold. Kør stavekontrol, fordel skrive- og læseopgaverne, så en anden, end den der har skrevet et afsnit, læser korrektur på det.
- Det er bedre at aflevere noget end intet.
- Der er afsat 1 undervisningsfri og 3/2 alm. undervisningsuger til opgaven (7/11-30/11 fraregnet 14/11-20/11, som er mellemuge, og kursusaktiveter på parallelkurser). Det svarer til ca. 40 timers arbejde. Brug dem struktureret og målrettet. Lav f.eks. en tidsplan, så I ikke taber overblikket over projektforløbet.