Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Uge(r)seddel 4 – gruppeopgave (rev. 1.1)

Torben Mogensen

Deadline 5. oktober

I denne periode skal I arbejde i grupper. Formålet er at arbejde med tupler, lister, mønstergenkendelse og afprøvning.

Opgaverne i denne uge er delt i øve- og afleveringsopgaver. Vi bruger forkortelsen "HR" for Hansen & Rischels bog "Functional Programming Using F#".

NB! Der skal for alle opgaver laves black-box testing som beskrevet i afsnit 10.2 i Jon Sporrings F# noter. Dette gælder også afleveringsopgaverne.

Øveopgaverne er:

4ø.1. HR: 4.1, 4.7, 4.8.

4ø.2. En tabel kan repræsenteres som en liste af lister, hvor alle listerne er lige lange.

Listen `[[1; 2; 3]; [4; 5; 6]]` repræsenterer for eksempel tabellen

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

i Lav en funktion `isTable : 'a list list -> bool`, der givet en liste af lister afgør, om det er en lovlig ikke-tom tabel, altså om alle listerne har ens længde, og at der er mindst en liste med mindst et element. Vink: Se `TuplesAndLists.pdf`.

ii Lav en funktion `firstColumn : 'a list list -> 'a list`, der tager en liste af lister og returnerer listen af førsteelementer i de indre lister. F.eks. skal `firstColumn [[1; 2; 3]; [4; 5; 6]]` returnere listen `[1; 4]`.
Hvis en eller flere af listerne er tomme, er `firstColumn` udefineret. Derfor er det i orden, hvis `fsharpi` giver advarsel om "incomplete pattern matches", og en tilsvarende fejlmeddelelse, hvis `firstColumn` kaldes med et argument, hvor en af listerne er tom.

iii Lav en funktion `dropFirstColumn : 'a list list -> 'a list list`, der tager en liste af lister og returnerer en liste af lister, hvor førsteelementerne i de indre lister er fjernet. F.eks. skal `dropFirstColumn [[1; 2; 3]; [4; 5; 6]]` returnere `[[2; 3]; [5; 6]]`.
Hvis en eller flere af listerne er tomme, er `dropFirstColumn` udefineret. Derfor er det i orden, hvis `fsharpi` giver advarsel om "incomplete pattern matches", og en tilsvarende fejlmeddelelse, hvis `dropFirstColumn` kaldes med et argument, hvor en af listerne er tom.

iv Lav en funktion `transpose : 'a list list -> 'a list list`, der *transponerer* en tabel. Transponering er spejling over diagonalen, så den transponerede tabel til den herover viste tabel er

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Kaldet `transpose [[1; 2; 3]; [4; 5; 6]]` skal altså returnere `[[1; 4]; [2; 5]; [3; 6]]`.
Det kan antages, at argumentet til `transpose` er en lovlig tabel, så advarsler om ufuldstændige mønstre er acceptable – hvis funktionen eller virker.
Bemærk, at `transpose (transpose t) = t`, hvis `t` er en tabel.
Vink: Brug funktionerne `firstColumn` og `dropFirstColumn`.

Afleveringsopgaven løses i grupper og er:

4g.1. HR: 4.11 og 4.15.

4g.2. Lav en funktion `removeDuplicates : 'a list -> 'a list when 'a : equality`, som fjerner duplikater i en liste. For eksempel skal kaldet `removeDuplicates [1; 2; 1; 3: 2]` give resultatet `[1; 2; 3]`. Bemærk, at den første forekomst af et givet elememt bevares, mens de øvrige forekomster fjernes. De bevarede elementer bevarer deres indbyrdes position.

Afleveringsopgaven skal afleveres som en f# fil med løsningerne for alle delopgaverne, og som kan køres med fsharpi. Indsæt delopgavernes nummer som kommentarer over løsningerne til disse. Husk at inkludere afprøvning (black-box testing) for alle funktioner, og dokumenter funktionerne med kommentarer som beskrevet i kapitel 7 af Jon Sporrings F# noter. Filen skal navngives efter følgende konvention: Hn-*fornavn*1.*efternavn*1-...-*fornavnm*.*efternavnm*-4g.fsx, hvor n er et tocifret holdnummer. Eksempel: `H07-Anders.And-Faetter.Guf-Georg.Gearloes-4g.fsx`.

God fornøjelse

Ugens nød #1

Vi vil i udvalgte uger stille særligt udfordrende og sjove opgaver, som interesserede kan løse. Det er helt frivilligt at lave disse opgaver, som vi kalder "Ugens nød", men der vil blive givet en mindre præmie til den bedste løsning, der afleveres i Absalon.

Ugens nød i denne uge omhandler en variant af spillet „Minestryger" (`http://da.wikipedia.org/wiki/Minestryger_%28spil%29`).

I vores variant er det for alle felter på forhånd kendt, hvor mange bomber, der i alt er i de otte nabofelter samt feltet selv. Det gælder både felter med og felter uden bomber. Det er samme variant, der blev brugt i dagsløbet i campusdagene.

Et eksempel på et sådant spil er

```
221
232
232
```

svarende til bombeplaceringen

```
000
110
001
```

hvor tomme felter er angivet med `0` og bomber med `1`.

Der er ikke altid en entydig løsning. For eksempel vil spillet

```
11
11
```

have fire løsninger:

```
01    10    00    00
00    00    01    10
```

Der er heller ikke altid løsninger, idet for eksempel

```
10
00
```

ikke har en løsning.

Vi repræsenterer et spil som en tabel af heltal, som alle ligger mellem 0 og 9. Løsninger repræsenteres som tabeller af heltal, der alle har værdi 0 eller 1.

Nød 1.1 funktion `minelaegger : int list list -> int list list`, som givet en *løsning* returnerer et spil. Det kan antages, at input er en liste af lister af tal, der alle er enten 0 eller 1.

For eksempel skal kaldet `minelaegger [[0; 0; 0]; [1; 1; 0]; [0; 0; 1]]` returnere spillet `[[2; 2; 1]; [2; 3; 2]; [2; 3; 2]]`.

Nød 1.2 Lav en funktion `minestryger : int list list -> int list list list`, som givet et spil finder alle løsninger. Det kan antages, at input er en liste af lister af tal mellem 0 og 9.

For eksempel skal kaldet `minestryger [[1; 1]; [1; 1]];;` returnere listen

```
[[[0; 1]; [0; 0]];
 [[1; 0]; [0; 0]];
 [[0; 0]; [0; 1]];
 [[0; 0]; [1; 0]]]
```

eller en permutation af denne liste, mens `minestryger [[1; 0]; [0; 0]]` skal returnere den tomme liste.

Det er ikke vigtigt, om løsningerne findes hurtigt, men hvis flere korrekte programmer indleveres, gives præmien til det hurtigste.

Der skal uploades både en LaTeX-fil, der beskriver fremgangsmåden, samt en fsx fil, der indeholder definitionerne af de to funktioner. Navngivningen af filerne er ikke vigtig. Der er oprettet en særlig opgave til ugens nød. Opgaven er individuel.

Endnu et par eksempelspil til afprøvning er angivet herunder.

22222	33333	44444	23232	23321
22222	33333	44444	24453	34421
22222	33333	44444	45543	45532
22222	33333	44444	34443	23321
22222	33333	44444	33222	12221