Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Uge(r)seddel 5 – individuel opgave
Revision 1.1 – tilføjet tekst markeret med rød

Torben Mogensen

Deadline 12. oktober

I denne periode skal I arbejde individuelt. Formålet er at arbejde med lister, mængder, og arrays, og specielt med biblioteksfunktioner, der arbejder på disse typer.

Opgaverne i denne uge er delt i øve- og afleveringsopgaver.

Øveopgaverne er:

ø5.1. Definer en funktion `reverseApply : 'a -> ('a -> 'b) -> 'b`, sådan at `reverseApply x f` returnerer resultatet af funktionsanvendelsen `f x`.

ø5.2. Forklar forskellen mellem typerne `int -> (int -> int)` og `(int -> int) -> int`, og giv et eksempel på en funktion af hver type.

ø5.3. Brug `List.filter` til at lave en funktion `evens : int list -> int list`, der returnerer de lige elementer i en liste.

ø5.4. Brug `List.map` og `reverseApply` (fra opgave ø5.1) til at lave en funktion `applylist : ('a -> 'b) list -> 'a -> 'b list`, der anvender en liste af funktioner på samme element for at returnere en liste af resultater.

ø5.5. HR: 5.1.

ø5.6. En snedig programmør definerer en sorteringsfunktion med definitionen

```
ssort xs = Set.toList (Set.ofList xs)
```

For eksempel giver `ssort [4; 3; 7; 2]` resultatet `[2; 3; 4; 7]`.

Diskuter, om programmøren faktisk er så snedig, som han tror.

ø5.7. Brug `Array.init` til at lave en funktion `squares: int -> int []`, sådan at kaldet `squares` n returnerer listen af de n første kvadrattal. For eksempel skal `squares 5` returnere arrayet `[|1; 4; 9; 16; 25|]`. Hvis `squares` gives et negativt tal, skal en fejlmeddelelse gives ved hjælp af funktionen `failwith`.

ø5.8. Brug `Array.init` og `Array.length` til at skrive en funktion `reverseArray : 'a [] -> 'a []`, der returnerer arrayet med elementerne i omvendt rækkefølge. For eksempel skal kaldet

```
printfn "%A" (reverseArray [|1..5|])
```

udskrive `[|5; 4; 3; 2; 1|]`.

ø5.9. Brug en while-løkke og overskrivning af array-elementer til at skrive en funktion `reverseArrayD : 'a [] -> unit`, som destruktivt opdaterer et array, så elementerne kommer i omvendt rækkefølge.

Sekvensen

```
let aa = [|1..5|]
reverseArrayD aa
printfn "%A" aa
```

skal altså udskrive `[|5; 4; 3; 2; 1|]`.

ø5.10. Brug `Array2D.init`, `Array2D.length1` og `Array2D.length2` til at lave en funktion `transpose : 'a [,] -> 'a [,]` som returnerer det transponerede argument, dvs. spejler det over diagonalen.

Afleveringsopgaven er:

i5.1. Brug funktionerne fra Tabel 5.1 i HR (side 94) til at definere en funktion `concat : 'a list list -> 'a list`, der sammensætter en liste af lister til en enkelt liste.

F.eks. skal `concat [[2]; [6; 4]; [1]]` give resultatet `[2; 6; 4; 1]`.

i5.2. Brug funktionerne fra Tabel 5.1 i HR (side 94) til at definere en funktion `gennemsnit : float list -> float option`, der finder gennemsnittet af en liste af kommatal, såfremt dette er veldefineret, og `None`, hvis ikke.

i5.3. Lav en funktion `arraySort : ('a [] -> 'a []) when 'a : comparison`, som givet et array returnerer en sorteret udgave af samme array. Lav din løsning uden brug af `<-` operatoren (altså funktionelt).

For eksempel skal sekvensen

```
let aa = [|1;7;5;2;1|]
let bb = arraySort aa
printfn "%A %A" aa bb
```

udskrive

```
[|1;7;5;2;1|] [|1;1;2;5;7|]
```

i5.4. Lav en funktion `arraySortD : ('a [] -> unit) when 'a : comparison`, som sorterer et array, sådan at elementerne i det oprindelige array overskrives med nye, sorterede elementer, altså imperativt.

For eksempel skal sekvensen

```
let aa = [|1;7;5;2;1|]
arraySortD aa
printfn "%A" aa
```

udskrive

```
[|1;1;2;5;7|]
```

Afleveringsopgaven skal afleveres som både LaTeX, den genererede PDF, samt en fsx fil med løsningen for hver delopgave, navngivet efter opgaven (f.eks. `i5-1.fsx`), som kan køres med fsharpi. Det hele samles i en zip-fil med navnekonventionen H*n-fornavn.efternavn*`-5i.zip`.

Tilføjet Tekst: LaTeX-rapporten skal kort beskrive de metoder, er er brugt i sorteringsfunktionerne.

God fornøjelse

Ugens nød 2

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. Opgaverne er individuelle.

Biblioteket `makeBMP.dll`

I "Filer" på Absalonsiden for kurset kan du finde filen `makeBMP.dll`, som er et bibliotek med funktioner til at læse og skrive BMP grafikfiler. Følgende funktioner er defineret:

`makeBMP.makeBMP : string -> int -> int -> (int*int -> int*int*int) -> unit`

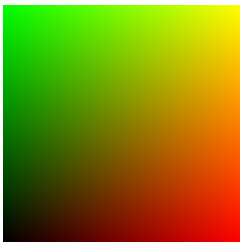
Tager et navn f, en bredde w, en højde h og en funktion c fra punkter til farver, og laver en BMP fil f`.bmp`, med $w \times h$ pixels, hvor farven i punkt (i, j) er defineret ved farven $c(i, j)$.

En farve er et RGB-tripel af tre tal mellem 0 og 255, hvor det første tal er den røde komponent, det andet tal er den grønne komponent, og det tredje tal er den blå komponent, så gul kan f.eks. defineres med triplen (255,255,0) og grå som (100,100,100).

Hvis f.eks. programfilen `tryBMP.fsx` består af linjen

```
makeBMP.makeBMP "test" 256 256 (fun (x,y) -> (x,y,0))
```

så vil kørsel af denne fil med kommandoen `fsharpi -r makeBMP.dll tryBMP.fsx` producere en fil `test.bmp` med følgende indhold:



`makeBMP.makeBMParray : string -> int -> int -> int*int*int [,] -> unit`

Fungerer ligesom `makeBMP.makeBMP`, men i stedet for at definere farverne med en funktion, defineres det af et $w \times h$ todimensionelt array af farver.

`makeBMP.readBMP : string -> int * int * (int*int -> (int*int*int))`

Tager et navn f og indlæser billedet i filen f`.bmp` og returnerer en tripel $(w, h\, c)$, hvor w og h er bredde og højde af billedet og c er en funktion fra punkter til farver.

`makeBMP.readBMParray : string -> int * int * ( (int*int*int) [,])`

Tager et navn f og indlæser billedet i filen f`.bmp` og returnerer en tripel $(w, h\, a)$, hvor w og h er bredde og højde af billedet og a er et todimensionalt array af farver.

Der er nogle begrænsninger på funktionerne:

- w og h må højest være 8192.
- Indlæsningsfunktionerne kan ikke læse alle former for BMP filer, men kan håndtere de mest almindelige.

Man kan konvertere forskellige billedeformater til og fra BMP med f.eks. IrfanView og Gimp. På Linux kan man endvidere skrive f.eks.

```
convert test.bmp test.png
```

for at konvertere fra BMP til PNG.

Opgaven

Du skal bruge de ovennævnte funktioner til at lave et flot billede. Der er følgende begrænsninger:

1. Billedfilen skal navngives *FornavnEfternavn*`.bmp` (med dit eget navn).
2. Billedet skal være eksakt 512×512 pixels.
3. Programmet, der laver billedet må bestå af højest 20 programlinjer, og ingen programlinje må være mere end 80 tegn lang.

   Blanke linjer og linjer, der kun indeholder kommentarer, tælles ikke som programlinjer i denne optælling.
4. Programmet må ikke bruge mere end to minutter til at køre på Torbens laptop, når det køres med kommandoen

   `fsharpi -r makeBMP.dll` *FornavnEfternavn*`.fsx`
5. Deadline er 14. oktober kl. 12.00. Opgaven uploades på den tilhørende opgaveside på Absalon.

Programmer, der ikke opfylder kravene, diskvalificeres uanset hvor flotte de genererede billeder er.

Billederne vurderes 100% subjektivt af et panel af undervisere og instruktorer. Kriterier er originalitet, æstetik, imponeringsfaktor, samt hvor godt, fremgangsmetoden er beskrevet. Hvis der indlæses billeder, vurderes det genererede billede ud fra, hvad det tilfører i forhold til de indlæste billeder.

Der skal uploades både en LaTeX-fil, der beskriver fremgangsmåden, en fil *FornavnEfternavn*`.fsx`, der indeholder programmet, og en BMP fil, der indeholder billedet. Hvis der indlæses billeder, skal disse vedlægges. Det hele samles i en zip-fil `N2-`*FornavnEfternavn*`.zip`