Programmering og Problemløsning
Datalogisk Institut, Københavns Universitet
Uge(r)seddel 6 – gruppeopgave
Revision 1.01 – der er rettet et par trykfejl og klarificeret rapportindhold

Torben Mogensen

Deadline 26. oktober

I denne periode skal I arbejde grupper. Formålet er at arbejde med sumtyper og endelige træer.

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

Øveopgaverne er:

Givet en sum-type til representation af ugedage:

```
type weekday = Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday
```

ø6.1. Lav en funktion `dayToNumber : weekday -> int`, der givet en ugedag returnerer et tal, hvor mandag skal give tallet 1, tirsdag tallet 2 osv.

ø6.2. Lav en funktion `nextDay : weekday -> weekday`, der givet en ugedag returnerer den næste dag, så mandag skal give tirsdag, tirsdag skal give onsdag, osv, og søndag skal give mandag.

Vi arbejder i de følgende opgaver (både øveopgaver og afleveringsopgaver) med en træstruktur til at beskrive geometriske figurer med farver. For at gøre det muligt at afprøve jeres opgaver, udleveres et simpelt bibliotek `makeBMP.dll`, der kan lave bitmapfiler. Biblioteket er beskrevet i Ugens Nød i ugeseddel 5. Her bruger vi kun funktionen

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

Det første argument er navnet på den ønskede bitmapfil (uden extension). Det andet argument er bredden af billedet i antal pixel, det tredje element er højden af billedet i antal pixel og det sidste argument er en funktion, der afbilder koordinater i billedet til farver. En farve er en tripel af tre tal mellem 0 og 255 (begge inklusive), der beskriver hhv. den røde, grønne og blå del af farven.

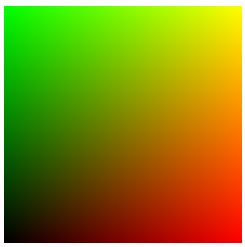
Koordinaterne starter med $(0, 0)$ i nederste venstre hjørne og $(w-1, h-1)$ i øverste højre hjørne, hvis bredde og højde er hhv. w og h. For eksempel vil en programfil `testBMP.fsx` med indholdet

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

kunne køres med kommandoen

```
fsharpi -r makeBMP.dll testBMP.fsx
```

og lave en billedfil med navnet `test.bmp`, der indeholder følgende billede:



Bemærk, at alle programmer, der bruger `makeBMP` skal køres eller oversættes med `-r makeBMP.dll` som en del af kommandoen.

Bemærk endvidere, at LaTeX ikke kan inkludere BMP-filer med `includegraphics`. Men man kan bruge diverse billedprogrammer (IrfanView, osv.) til at konvertere BMP til PNG, som `includegraphics` godt kan håndtere.

I Linux kan man konvertere billeder med kommandoen `convert`. Hvis vi har billedfilen `test.bmp`, vil kommandoen `convert test.bmp test.png` lave en tilsvarende billedfil `test.png` i PNG-formatet.

Vi repræsenterer geometriske figurer med følgende datastruktur:

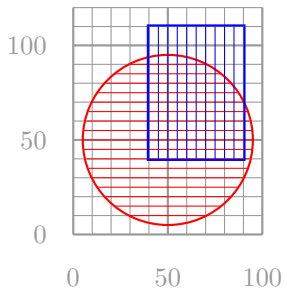
```
type point = int * int // a point (x, y) in the plane
type colour = int * int * int  // (red, green, blue), 0..255 each

type figure =
        | Circle of point * int * colour
          // defined by center, radius, and colour
        | Rectangle of point * point * colour
          // defined by bottom-left corner, top-right corner, and colour
        | Mix of figure * figure
          // combine figures with mixed colour at overlap
```

For eksempel kan man lave følgende funktion til at finde farven af en figur i et punkt. Hvis punktet ikke ligger i figuren, returneres `None`, og hvis punktet ligger i figuren, returneres `Some` c, hvor c er farven.

```
// finds colour of figure at point
let rec colourAt (x,y) figure =
  match figure with
  | Circle ((cx,cy), r, col) ->
      if (x-cx)*(x-cx)+(y-cy)*(y-cy) <= r*r
         // bruger Pythagoras sætning til at finde afstand til centrum
      then Some col else None
  | Rectangle ((x0,y0), (x1,y1), col) ->
      if x0<=x && x <= x1 && y0 <= y && y <= y1  // indenfor hjørnerne
      then Some col else None
  | Mix (f1, f2) ->
      match (colourAt (x,y) f1, colourAt (x,y) f2) with
      | (None, c) -> c  // overlapper ikke
      | (c, None) -> c  // ditto
      | (Some (r1,g1,b1), Some (r2,g2,b2)) ->
           Some ((r1+r2)/2, (g1+g2)/2, (b1+b2)/2)  // gennemsnitsfarve
```

Bemærk, at punkter på cirklens omkreds og rektanglens kanter er med i figuren. Farver blandes ved at lægge dem sammen og dele med to, altså finde gennemsnitsfarven.

ø6.3. Lav en figur `o61 : figure`, der består af en rød cirkel med centrum i (50,50) og radius 45, samt en blå rektangel med hjørnerne (40,40) og (90,110), som illustreret med denne tegning:



Hvor vi dog har brugt skravering i stedet for udfyldende farver.

ø6.4. Brug `makeBMP` og `colourAt` til at lave en funktion

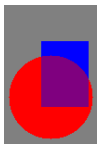
```
makePicture : string -> figure -> int -> int -> unit
```

sådan at kaldet `makePicture` *`filnavn figur b h`* laver en billedfil ved navn *`filnavn`*`.bmp` med et billede af *`figur`* med bredde *`b`* og højde *`h`*.

På punkter, der ingen farve har (jvf. `colourAt`), skal farven være grå (som defineres med RGB-værdien (128,128,128)).

Du kan bruge denne funktion til at afprøve dine opgaver.

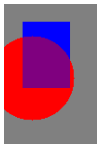
ø6.5. Lav med `makePicture` en billedfil med navnet `o63.bmp` og størrelse 100×150 (bredde 100, højde 150), der viser figuren `o61` fra opgave ø6.1.

Resultatet skulle gerne ligne 

ø6.6. Lav en funktion `checkFigure : figure -> bool`, der undersøger, om en figur er korrekt: At radiusen i cirkler er ikke-negativ, at nederste venstre hjørne i en rektangel faktisk er nedenunder og til venstre for det øverste højre hjørne (bredde og højde kan dog godt være 0), og at farvekompenterne ligger mellem 0 og 255.

Vink: Lav en hjælpefunktion `checkColour : colour -> bool`.

ø6.7. Lav en funktion `move : figure -> int * int -> figure`, der givet en figur og en vektor flytter figuren langs vektoren.

Hvis man kalder `makePicture "moveTest"(move o61 (-20,20)) 100 150`, shulle det gerne lave en billedfil `moveTest.bmp` med indholdet 

ø6.8. Lav en funktion `boundingBox : figure -> point * point`, der givet en figur finder hjørnerne (bund-venstre og top-højre) for den mindste akserette rektangel, der indeholder hele figuren.

`boundingBox o61` skulle gerne give `((5, 5), (95, 110))`.

Afleveringsopgaven er:

g6.1. Givet typen for ugedage øverst på denne ugeseddel, lav en funktion `numberToDay : int -> weekday option`, sådan at `numberToDay` n returnerer `None`, hvis n ikke ligger i intervallet $1 \ldots 7$, og returnerer `Some` d, hvor d er den til n hørende ugedag, hvis n ligger i intervallet $1 \ldots 7$.

Det skulle gerne gælde, at `numberToDay (dayToNumber` d`)` $\rightsquigarrow$ `Some` d for alle ugedage d.

Til de følgende opgaver udvider vi typen `figure`:

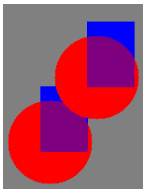
```
type figure =
        | Circle of point * int * colour
          // defined by center, radius, and colour
        | Rectangle of point * point * colour
          // defined by bottom-left corner, top-right corner, and colour
        | Mix of figure * figure
          // combine figures with mixed colour at overlap
        | Twice of figure * (int * int)
          // overlays figure with copy of self moved by vector
```

hvor `Twice` f (x, y) gentager figuren f i to kopier, hvor det første kopi af f ligger normalt, og andet kopi er forskudt med vektoren (x, y). Hvis de to kopier overlapper, ligger andet kopi ovenpå det første. Der skal altså ikke blandes farver mellem de to kopier (men internt i hvert kopi kan `Mix` bruges til at blande farver ved overlap). Bemærk, at x og y kan være negative.

g6.2. Lav en figur `g61 : figure`, som består af to kopier af figur `o61`, hvor det andet er forskudt med vektoren $(50, 70)$.

g6.3. Udvid funktionen `colourAt` til at håndtere udvidelsen.

g6.4. Lav en fil `g63.bmp`, der viser figuren `g61` i et 150×200 bitmap.

Resultatet skulle gerne ligne 

g6.5. Udvid funktionerne `checkFigure` og `boundingBox` fra øvelsesopgaverne til at håndtere udvidelsen.

`boundingBox g61` skulle gerne give `((5, 5), (145, 180))`.

Der skal laves black-box testing og in-code dokumentation af funktionerne.

Afleveringsopgaven skal afleveres som både LaTeX, genererede billedfiler, den genererede PDF, samt en fsx fil med løsningen for hver delopgave, navngivet efter opgaven (f.eks. `g6-1.fsx`), som kan oversættes med fsharpc, og hvis resultat kan køres med mono. Det hele samles i en zip-fil med sædvanlig navnekonvention (se tidligere ugesedler).

LaTeX-rapporten skal vise de billedfiler, der bliver lavet i opgaverne, og forklare ikke-oplagte designvalg i løsningerne af opgaverne.

God fornøjelse

Ugens nød 3

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.

Denne nød omhandler udvidelser/ændringer til datastrukturen for geometriske funktioner, som er introduceret i øvelses og afleveringsopgaverne herover. Vi erstatter figurdatatypen med:

```
type figure =
        | Circle of point * int * (colour * colour)
          // defined by center, radius, and centre/edge colours
        | Rectangle of point * point * (colour * colour * colour * colour)
          // defined by bottom-left corner, top-right corner,
          // and colours for each corner in the order (bl, br, tr, tl)
        | Mix of figure * figure
          // combine figures with mixed colour at overlap
        | Twice of figure * (int * int)
          // overlays figure with copy of self moved by vector
        | Ellipse of point * point * int * (colour * colour)
          // defined by two focal points and length of the great axis
          // with colours at each focal point
        | Triangle of point * point * point * (colour * colour * colour)
          // defined by three vertices and colours at these
```

Vi har dels udvidet figurerne med ellipser og trekanter, og dels ændret farverne fra at være ensartede henover en figur til at være glidende overgange:

- I en cirkel skal farven variere lineært fra centrum til kant mellem de to specificerede farver.
- I en rektangel eller trekant skal farverne variere, sådan at farverne i hjørnerne er som angivet, og farverne skifter glidende mellem disse henover det indre. Angiv hvordan du definerer farveblandingen i et punkt.

  Et kvadrat med farverne sort, rød, gul og grøn i hjørnerne skulle gerne ligne billedet på side 1 i denne ugeseddel (men behøver ikke at være eksakt ligedan).
- I en ellipse skal farverne i brændpunkterne være som angivet, og farven i resten af ellipsen skal afhænge af afstandende til de to brændpunkter. Hvis de to afstande er lige store, skal farven blandes ligeligt, ellers skal farven afhænge af forholdet mellem de to afstande, sådan at farveovergangen er glidende. Angiv formlen for farve baseret på forholdet mellem afstandende.
- En trekant er korrekt, så længe farverne i hjørnerne er korrekt.
- En ellipse er korrekt, hvis de to brændpunkter er forskellige, afstanden mellem brændpunkterne er skarpt mindre end storaksens længde, og de to farver er korrekte.

Definer versioner af `colourAt`, `checkFigure`, og `boundingBox`, der passer til den nye udgave af `figure`, og følger de ovenstående retningslinjer.

Lav figurer, der afprøver glidende farveovergange samt korrekthed af ovenstående funktioner. Lav en rapport, der beskriver designvalg, blandt andet til farveovergange og til at finde bounding-box for en ellipse. Bemærk, at en tæt bounding-box for en ellipse kan kræve ikke-heltallige koordinater. Disse skal rundes ned/op, sådan at `boundingBox` returnerer den mindste kasse med heltallige koordinater, der indeholder ellipsen.

Upload en zip-fil med LaTeX-fil, PDF og .fsx filer. Deadline er 28. oktober kl. 12.00 middag.

Besvarelserne bliver bedømt på korrekthed, elegance, og på hvor godt rapporten beskriver og begrunder designvalgene.