Reeksamensopgaver 2018-19

Jon Sporring

April 8, 2019

1 Opgave 1

I denne opgave skal du arbejde med polynomier og det imperative programmeringsparadigmet i F#.

Betragt et n'te grads polynomium,

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n. \tag{1}$$

Dette polynomium kan repræsenteres som en array af koefficienter. F.eks. kan polynomiet $1 + 3.2x - 2x^3$ repræsenteres som følgende F# array `[|1.0; 3.2; 0.0; -2.0|]`.

Løs følgende opgaver:

- Skriv en funktion med type `mulByConst : c:float -> p:float [] -> float []`, som ganger polynomiet `p` repræsenteret som en array med konstanten `c` og returnerer resultatet som et nyt polynomium igen repræsenteret som en array. F.eks., skal `mulByConst 3.0 [|1.0; 3.2; 0.0; -2.0|]` returnere `[|3.0; 9.6; 0.0; -6.0|]`.
- Skriv en funktion med type `mulByX : p:float [] -> float []`, som ganger polynomiet `p` med `x`. F.eks., skal `mulByX [|1.0; 3.2; 0.0; -2.0|]` returnere `[|0.0; 1.0; 3.2; 0.0; -2.0|]`.
- Skriv en funktion med type `add : p:float [] -> q:float [] -> float []`, som adderer polynomierne `p` og `q` med hinanden uden rekursion men vha. en eller flere `for`-løkker. F.eks., skal `add [|1.0; 3.2; 0.0; -2.0|] [|2.0; 1.3|]` returnere `[|3.0; 4.5; 0.0; -2.0|]`.
- Skriv en funktion med type `mul : p:float [] -> q:float [] -> float []`, som ganger polynomierne `p` og `q` med hinanden vha. rekursion. F.eks., skal `mul [|1.0; 3.2; 0.0; -2.0|] [|2.0 1.3|]` returnere `[|2.0; 7.7; 4.16; -4; -2.6|]`. Løsningen skal gøre brug af ovenstående funktioner og gøre brug af følgende rekursion:

  Base: $0q(x) = 0$,

  Rekursion: $\left(a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n\right) q(x) = a_0 q(x) + x\left(\left(a_1 + a_2 x + \ldots + a_n x^{n-1}\right) q(x)\right)$,

  hvor $q(x)$ er et vilkårligt polynomium.

2 Opgave 2

I denne opgave skal du arbejde med polynomier og funktionsprogrammeringsparadigmet i F#.

Betragt et n'te grads polynomium,

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n. \tag{2}$$

Dette polynomium kan repræsenteres som en liste af koefficienter. F.eks. kan polynomiet $1 + 3.2x - 2x^3$ repræsenteres som F# listen `[1.0; 3.2; 0.0; -2.0]`.

Løs følgende opgaver:

- Skriv en funktion med type `mulByConst : c:float -> p:float list -> float list`, som ganger polynomiet `p` repræsenteret som en liste med konstanten `c` og returnerer resultatet som et nyt polynomium igen repræsenteret som en liste. F.eks., skal `mulByConst 3.0 [1.0; 3.2; 0.0; -2.0]` returnere `[3.0; 9.6; 0.0; -6.0]`.
- Skriv en funktion med type `mulByX : p:float list -> float list`, som ganger polynomiet `p` med `x`. F.eks., skal `mulByX [1.0; 3.2; 0.0; -2.0]` returnere `[0.0; 1.0; 3.2; 0.0; -2.0]`.
- Skriv en funktion med type `add : p:float list -> q:float list -> float list`, som adderer polynomierne `p` og `q` med hinanden vha. rekursion. F.eks., skal `add [1.0; 3.2; 0.0; -2.0] [2.0; 1.3]` returnere `[3.0; 4.5; 0.0; -2.0]`.
- Skriv en funktion med type `mul : p:float list -> q:float list -> float list`, som ganger polynomierne `p` og `q` med hinanden. F.eks., skal `mul [1.0; 3.2; 0.0; -2.0] [2.0 1.3]` returnere `[2.0; 7.7; 4.16; -4; -2.6]`. Løsningen skal gøre brug af ovenstående funktioner og gøre brug af følgende rekursion:

  Base: $0q(x) = 0$,

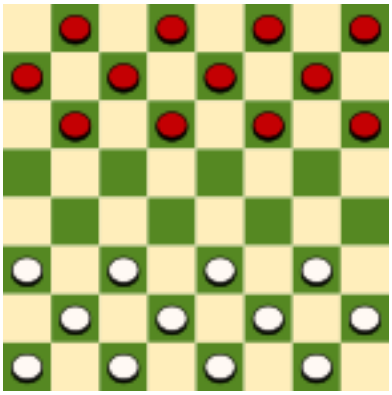
  Rekursion: $\left(a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n\right) q(x) = a_0 q(x) + x\left(\left(a_1 + a_2 x + \ldots + a_n x^{n-1}\right) q(x)\right)$,

  hvor $q(x)$ er et vilkårligt polynomium.

3 Opgave 3

I denne opgave skal du arbejde med brætspillet Dam og objektorienteret programmering og design i F#.

Spillet er som følger:

> Dam spilles af 2 spillere på et ternet bræt med lyse og mørke felter. Ved start får hver spiller 12 brikker i hhv. lys og mørk farve, og brikkerne anbringes i de 3 første rækker på de mørke felter over for hinanden, som vist nedenfor.
>
> 
>
> Der spilles kun på de mørke felter, og brikken flyttes skråt fremad et felt ad gangen.
>
> Modstanderens brikker slås, hvis disse støder op til ens egne med et tomt felt skråt bagved. Flere brikker kan slås i samme træk, bare der hele tiden er et tomt felt bag den brik, som erobres. Man kan skifte retning for hver brik man har slået, så længe man bevæger sig fremad.
>
> Lykkes det at komme frem til modstanderens bageste række, opnår man en dam. Denne brik markeres ved at placere en af ens slagne brikker ovenpå. En dam kan flyttes på skrå lige så mange felter frem eller tilbage, som der er tomme felter, og slå modstanderens brikker som tidligere.
>
> Spilleren, der har slået alle modstanderens brikker, eller har lukket dem inde, så de ikke kan flyttes, har vundet

I denne opgave skal du lave en skitse til at program, hvor 2 spillere kan spille Dam med hinanden på en computer. Dertil skal du lave følgende opgaver:

- Identificer nyttige klasser og metoder vha. navne- og udsagnsordsmetoden
- Tegn et UML diagram til et udkast til et design for spillet.
- Skriv et mockup program i F#, som implementerer dit design med alle de klasser og metoder, som dit design indeholder og med korte kommentarer om de væsentlige opgaver, som hver klasse og metode har i programmet.