**Bachelor PRD** **IT University, E2009**

# Skriftlig eksamen, Programmer som Data
# Onsdag 6. januar 2010

Dette eksamenssæt har 5 sider. Tjek med det samme at du har alle siderne.

Eksamens varighed er 4 timer.

Der er fire spørgmål. For at få fuldt point skal du besvare alle delspørgsmålene tilfredsstillende. Du må bruge alle bøger, forelæsningsnoter, forelæsningsplancher, opgavesæt, opgavebesvarelser, lommeregnere og så videre under eksamen, men ingen computere (heller ikke mobiltelefoner, PDA, iPod eller lignende) som kan udføre programmer i F# eller C# eller Java, eller som kan kommunikere med andre enheder.

Hvis et spørgsmål kræver at du definerer en bestemt funktion, så må du gerne bruge den funktion i efterfølgende spørgsmål, også selv om du ikke selv har defineret den.

Hvis et spørgsmål kræver at du definerer en bestemt funktion, så må du gerne definere alle de hjælpefunktioner du vil, men du skal definere den ønskede funktion så den har netop den type og det resultat som spørgsmålet kræver.

## Spørgsmål 1 (20 %): Regulære udtryk og automater

Betragt dette regulære udtryk over alfabetet $\{b, c\}$:

$$c(b|c)^*c$$

### Spørgsmål 1.1

Giv nogle eksempler på strenge der beskrives af dette regulære udtryk. Giv en uformel beskrivelse af sproget (mængden af alle strenge) der beskrives af dette regulære udtryk.

### Spørgsmål 1.2

Konstruer og tegn en ikke-deterministisk endelig automat ("nondeterministic finite automaton", NFA) der svarer til det regulære udtryk. Husk at angive starttilstand og accepttilstand(e).

### Spørgsmål 1.3

Konstruer og tegn en deterministisk endelig automat ("deterministic finite automaton", DFA) der svarer til det regulære udtryk, som i Mogensens noter. Husk at angive starttilstand og accepttilstand(e).

### Spørgsmål 1.4

Betragt følgende uendelige mængde af strenge over alfabetet $\{a, b\}$:

```
a
b
aa
bbb
aaa
bbbbb
...
```

Angiv et regulært udtryk der beskriver denne mængde af strenge.

## Spørgsmål 2 (30 %): F#

Opgaverne i dette spørgsmål handler om funktioner i F# til at simulere kast med terninger, for eksempel i forbindelse med et (bræt)spil eller i fredagsbaren. Et enkelt terningkast med en `s`-sidet terning kan udføres af funktionen `rollOne`, defineret således:

```
let rnd = new System.Random();;

let rollOne s = rnd.Next(s)+1;;
```

For eksempel vil et kald af `rollOne 6` svare til kast med en 6-sidet terning og hvert nyt kald vil returnere et tilfældigt af heltallene 1, 2, 3, 4, 5 eller 6. Et kald af `rollOne 20` svarer til kast med en 20-sidet terning og hvert nyt kalda vil returnere et tilfældigt af tallene 1, 2, . . . , 20.

### Spørgsmål 2.1

Definer en F# funktion `rollMany : int -> int -> int list` til at kaste mange terninger på en gang. Kaldet `rollMany n s` skal returnere en liste af længde `n` hvor hvert element er resultatet af et kast med en `s`-sidet terning. For eksempel kan det være at `rollMany 4 6` returnerer listen `[3; 2; 6; 5]`.

### Spørgsmål 2.2

Definer en F# funktion `rollSum : int -> int -> int` til at kaste mange terninger på en gang og returnere summen af udfaldene. Kaldet `rollSum n s` skal således returnere summen af udfaldene ved `n` kast med en `s`-sidet terning. For eksempel kan det være at `rollSum 4 6` returnerer summen `16`.

### Spørgsmål 2.3

Definer en F# funktion `count : (int -> bool) -> int list -> int` til at tælle hvor mange elementer i en talliste der opfylder en given betingelse. Kaldet `count p xs` skal således tælle hvor mange elementer `x` der findes i listen `xs` hvor `p(x)` er sand. For eksempel skal `count (fun x -> x>4) [3; 2; 6; 5]` returnere `2` fordi der er to tal større end 4 i den givne liste.

### Spørgsmål 2.4

Definer en F# funktion `smallest : int -> int list -> int list` til at returnere de mindste elementer fra en liste. Kaldet `smallest n xs` skal således returnere de `n` mindste elementer fra listen `xs`. Hvis der er færre end `n` elementer i `xs` skal hele listen returneres.

Vink: Den indbyggede funktion `List.sort : int list -> int list` kan bruges til at sortere en liste i voksende orden. For eksempel vil `List.sort [3; 2; 6; 5]` returnere `[2; 3; 5; 6]`.

### Spørgsmål 2.5

Definer en F# funktion `largest : int -> int list -> int list` til at returnere de største elementer fra en liste. Kaldet `largest n xs` skal således returnere de `n` største elementer fra listen `xs`. Hvis der er færre end `n` elementer i `xs` skal hele listen returneres.

# Spørgsmål 3 (20 %): Abstrakt syntaks og stakmaskine for Micro-C

Den abstrakte syntaks for kursets sprog Micro-C fremgår af afsnit 7.6 af *Programming Language Concepts for Software Developers* og af slides fra forelæsning 7.

For eksempel repræsenteres Micro-C udtrykket `x+1` således i abstrakt syntaks:

```
Prim2 ("+",Access (AccVar "x"),CstI 1)
```

mens tildelingsudtrykket `y = x + 1` repræsenteres således:

```
Assign (AccVar "y",Prim2 ("+",Access (AccVar "x"),CstI 1))
```

## Spørgsmål 3.1

Vis hvordan tildelingsudtrykket `x = x + 2 * y` repræsenteres som abstrakt syntaks.

Vis dernæst hvilket udtryk der repræsenteres af denne abstrakt syntaks:

```
Prim2 ("*",CstI 3, Prim2 ("+",Access (AccVar "x"),Access (AccVar "y")))
```

## Spørgsmål 3.2

Vis hvordan denne ordre ("statement") repræsenteres i Micro-C abstrakt syntaks (af type `stmt`):

```
while (x>0)
  x = x - 1;
```

Vis dernæst hvordan denne ordre repræsenteres i Micro-C abstrakt syntaks:

```
while (arr[i]!=0)
  i = i + 1;
```

## Spørgsmål 3.3

En stakmaskine, der kan bruges til at udføre oversatte Micro-C programmer, er beskrevet i afsnit 8.2 af *Programming Language Concepts for Software Developers* og i slides fra forelæsning 8.

Betragt denne sekvens af stakmaskineinstruktioner (der er resultatet af at oversætte et Micro-C udtryk):

```
CSTI 5; CSTI 3; SUB; CSTI 4; CSTI 6; MUL; ADD
```

Vis skridtene i udførelse af dette program, herunder stakkens indhold efter hvert skridt. Antag at stakken er tom til at begynde med.

## Spørgsmål 3.4

Skitser hvilken stakmaskinkode følgende Micro-C ordre kan oversættes til, for eksempel med oversætteren gennemgået i kapitel 8 af noterne og i forelæsning 8:

```
while (x>0)
  x = x - 1;
```

Du kan antage at `x` er en lokal variabel med offset 0. Brug symbolske labels (for eksempel L1, L2) i stedet for absolutte kodeadresser.

## Spørgsmål 4 (30 %): Abstrakt syntaks, parsing og tjek af udtryk

I vejledninger til brætspil bruger man gerne notationen `d`$s$ til at beskrive at der kastes en $s$-sidet terning, og $n$`d`$s$ til at beskrive at der kastes $n$ styk $s$-sidede terninger. Bogstavet `d` står for "die", dvs. terning. For eksempel betyder `d6` at der skal kastes en 6-sidet terning, og `4d6` betyder at der skal kastes 4 styk 6-sidede terninger. Kast med én 6-sidet terning kan altså både beskrives med `d6` og med `1d6`; de to udtryk betyder det samme.

Der findes mange varianter af terningkast. Man kan for eksempel kaste fire terninger og kun betragte de tre største udfald; eller man kan kaste nogle terninger og tælle hvor mange af dem der viser 4 eller flere øjne, og så videre.

Derfor betragter vi et sprog af "terningkastudtryk" med følgende slags udtryk:

| **Udtryk** | **Betydning** | **Størrelse** $\lvert e\rvert$ | **Eksempel** |
|---|---|---|---|
| $n$ | Heltalskonstanten $n$ | 1 | `4` |
| `d`$s$ | Et kast med en $s$-sidet terning | 1 | `d20` |
| $n$`d`$s$ | $n$ kast med $s$-sidede terninger | $n$ | `4d6` |
| `sum` $e$ | Summen af udfaldene i kastet $e$ | 1 | `sum 4d6` |
| `largest` $k$ `of` $e$ | De $k$ største af udfaldene i kastet $e$ | $k$ | `largest 2 of 4d6` |
| `smallest` $k$ `of` $e$ | De $k$ mindste af udfaldene i kastet $e$ | $k$ | `smallest 2 of 4d6` |
| `count` $k$ `<` $e$ | Antal udfald i $e$ som er større end $k$ | 1 | `count 4 < 4d6` |
| `count` $k$ `>` $e$ | Antal udfald i $e$ som er mindre end $k$ | 1 | `count 4 > 4d6` |
| $e_1$ `+` $e_2$ | Summen af kastene $e_1$ og $e_2$ | $\lvert e_1\rvert$ | `4d10 + 4d6` |
| `(`$e$`)` | Samme som $e$ | $\lvert e\rvert$ | `(4d10)` |

Resultatet af et terningkastudtryk $e$ er en liste af tal. Størrelsen af resultatet (dvs. længden af listen) betegnes $|e|$ og fremgår af tredje kolonne ovenfor. For eksempel har resultatet af `4d6` størrelse 4, mens resultatet af `sum 4d6` har størrelse 1; summen er jo et enkelt tal. Den angivne størrelse $k$ af `largest` og `smallest` gælder for velformede udtryk; dette har kun betydning for delspørgsmål 4.4 som forklarer dette nærmere.

Meningen med sproget er at man kan skrive sammensatte terningkastudtryk såsom dette:

```
sum (largest 3 of 5d10)
```

der betyder "kast 5 styk 10-sidede terninger og find summen af de 3 største udfald", eller dette:

```
(count 3 < 5d6) + d6
```

der betyder "kast 5 styk 6-sidede terninger, tæl hvor mange af dem der viser mere end 3 øjne, og læg dertil resultatet af at kaste en enkelt 6-sidet terning".

Den abstrakte syntaks for terningkastsproget ser sådan ud:

```
type comp =
  | Lt
  | Gt
type expr =
  | CstI of int                (* Integer constant      *)
  | Roll of int * int          (* d6 or 1d6 or 2d10 ... *)
  | Sum of expr                (* sum e                 *)
  | Largest of int * expr      (* largest k of e        *)
  | Smallest of int * expr     (* smallest k of e       *)
  | Count of int * comp * expr (* count k >< e          *)
  | Plus of expr * expr        (* e + e                 *)
```

### Spørgsmål 4.1

Vis udtrykket svarende til den abstrakte syntaks `Plus(CstI 10, Sum(Roll(4, 6)))`.

Vis dernæst hvordan udtrykket `sum 4d6 + (count 2 < 3d10)` ser ud i abstrakt syntaks.

### Spørgsmål 4.2

Antag at disse token-erklæringer findes i en `fsyacc` parserspecifikation (`.fsy` fil) for terningkastsproget:

```
%token <int> CSTINT
%token LPAR RPAR GT LT PLUS ROLL
%token COUNT LARGEST OF SMALLEST SUM
```

Token `CSTINT` repræsenterer en positiv heltalskonstant, og de øvrige tokens svarer til disse input:

```
(  )  >  <  +  d
count  largest  of  smallest  sum
```

Skriv regeldelen af parserspecifikationen for terningkastsproget. Du behøver ikke skrive de semantiske aktioner i dette spørgsmål. Regeldelen kunne have denne form:

```
Expr:
    CSTINT                              { ... }
  | ...                                 { ... }
;
```

### Spørgsmål 4.3

Udvid parserspecifikationen fra foregående delspørgsmål med semantiske aktioner indeholdt i `{ ... }`, sådan at parseren konstruerer abstrakt syntaks svarende til den konkrete syntaks.

### Spørgsmål 4.4

Skriv en F# funktion `size : expr -> int` der returnerer størrelsen af resultatet af et terningkastudtryk, således at kaldet `size` $e$ returnerer størrelsen $|e|$, der er defineret i tabellen først i dette spørgsmål.

For eksempel skal `size (Roll(4, 6))` give 4, og `size (Sum(Roll(4, 6)))` skal give 1.

Funktionen `size` skal desuden tjekke at terningkastudtrykket $e$ er *velformet* ifølge disse regler:

- Resultatet af `largest` $k$ `of` $e$ har størrelse $k$ og er kun velformet hvis $e$ har størrelse $k$ eller mere. For eksempel er `largest 3 of 4d6` velformet, men det er `largest 5 of 4d6` ikke: man kan jo ikke vælge de 5 største af 4 tal.
- Samme regel gælder for `smallest` $k$ `of` $e$.
- Resultatet af additionen $e_1 + e_2$ har samme størrelse som resultatet af $e_1$, og additionen er kun velformet hvis disse resultaterne af $e_1$ og $e_2$ har samme størrelse, altså hvis $|e_1| = |e_2|$.

Hvis udtrykket `e` ikke overholder reglerne skal funktionskaldet `size e` kaste en exception, for eksempel ved at kalde den indbyggede funktion `failwith "size error"`.