Norsk

Universitetet i Bergen
Det matematisk-naturvitenskapelige fakultet

Eksamen i	:	INF-122 Funksjonell programmering
Dato	:	14 februar 2020
Tid	:	9:00 – 12:00
Antall sider	:	4 (inkl. vedlegg)
Tillatte hjelpemidler	:	Ingen

- Prosentsatsene angir *kun omtrentlig* vekting ved sensur og forventet tidsforbruk.
- Løsninger av delproblemer som du ikke har besvart kan antas gitt dersom de trenges i andre delproblemer.
- Programmer, og angi typen til, alle hjelpefunksjoner som du selv innfører. Din kode skal ikke forutsette andre funksjoner enn de som er tilgjengelige fra standard `Prelude`.

1 Velg/angi riktige svar **(25%)**

1.1. Evaluering av `filter even (map (*2) [1..5])` gir:
(a) `[2,4]` (c) `[2,6,10]`
(b) `[4,8]` (d) `[2,4,6,8,10]`

1.2. Hva blir resultatet av å evaluere `take 5 nats` med hver av følgende definisjoner?
(a) `nats = 0:1:tail nats` (c) `nats = 0:map (+1) nats`
(b) `nats = 0:tail nats` (d) `nats = map (+1) [0..]`

1.3. Hva blir resultatet av å evaluere `concat ["ab","cd","","efg"]` med hver av følgende definisjoner?
(a) `concat xss = [x | x<-xss]` (c) `concat xss = concat (tail xss)`
(b) `concat xss = [x | xs<-xss, x<-xs]` (d) `concat xss = map (++) xss`

1.4. Funksjon `apply` definert ved `apply f x = f x` har typen:
(a) `a -> b -> c` (c) `a -> (b -> a) -> b`
(b) `(a -> b) -> a -> b` (d) `a -> b -> (a -> b)`

2 Matrisemultiplikasjon **(20%)**

To $n \times n$ matriser multipliseres ved følgende formel ($x_{r,k}$ er tallet i r-te rad og k-te kolonne):

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,n} \\ b_{2,1} & b_{2,2} & \dots & b_{2,n} \\ \dots & \dots & \dots & \dots \\ b_{n,1} & b_{n,2} & \dots & b_{n,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \dots & \dots & \dots & \dots \\ c_{n,1} & c_{n,2} & \dots & c_{n,n} \end{bmatrix},$$

der, for $1 \leq r \leq n$ og $1 \leq k \leq n$, er tallet $c_{r,k}$ – i r-te rad og k-te kolonne av resultatmatrisen – definert ved følgende multiplikasjon av r-te rad fra første matrisen med k-te kolonne fra andre matrisen (*, + er vanlig multiplikasjon og addisjon):

$$c_{r,k} = [a_{r,1}, a_{r,2}, ..., a_{r,n}] \otimes \begin{bmatrix} b_{1,k} \\ b_{2,k} \\ ... \\ b_{n,k} \end{bmatrix} = a_{r,1} * b_{1,k} + a_{r,2} * b_{2,k} + ... + a_{r,n} * b_{n,k}.$$

For eksempel,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} (1*3+2*5) & (1*4+2*6) \\ (3*3+4*5) & (3*4+4*6) \end{bmatrix} = \begin{bmatrix} 13 & 16 \\ 29 & 36 \end{bmatrix}$$

Vi representerer en $n \times n$ matrise som en liste med n lister, tilsvarende rader i matrisen. F.eks. matriser $\left[\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right]$ og $\left[\begin{smallmatrix} 3 & 4 \\ 5 & 6 \end{smallmatrix}\right]$ representeres ved lister $[[1,2],[3,4]]$ og $[[3,4],[5,6]]$. Programmer følgende funksjoner (og eventuelle hjelpefunksjoner):

2.1. `row::[[Int]] -> Int -> [Int]` – `row m r` returner r-te rad fra matrisen m, f.eks. `row [[1,2],[3,4]] 2 = [3,4]`.

2.2. `col::[[Int]] -> Int -> [Int]` – `col m k` returner k-te kolonne fra matrisen m, f.eks. `col [[1,2],[3,4]] 1 = [1,3]`.

2.3. `cols::[[Int]] -> [[Int]]` – returnerer liste med kolonner for matrisen, f.eks. `cols [[1,2],[3,4]] = [[1,3],[2,4]]`.

2.4. `mult::[[Int]]->[[Int]]->[[Int]]` – gitt to $n \times n$ matriser, returnerer resultatet av deres multiplikasjon, f.eks., `mult [[1,2],[3,4]] [[3,4],[5,6]] = [[13,16],[29,36]]`.

3 IO og postfiksuttrykk (35%)

Vi betrakter følgende grammatikk for aritmetiske uttrykk i postfiks notasjon:

E ::= Pos | E E * | E E + | E E –
Pos ::= Digit | DigitPos
Digit ::= 0 | 1 | ... 8 | 9

Pos er ikke-terminal symbol for ikke-negative heltall (altså, uten noen minus tegn foran). Et uttrykk evalueres ved at en binær operator anvendes på to uttrykk til venstre for den, med uttrykket lenger til venstre som det første argumentet, f.eks.:

(a) "11 2 3 * + 4 –" tilsvarer (11 + (2 * 3)) – 4
(b) "11 2 * 3 + 4 –" tilsvarer ((11 * 2) + 3) – 4
(c) "4 11 2 3 * + –" tilsvarer 4 – (11 + (2 * 3))

Postfiks uttrykk evalueres online, mens de leses fra venstre til høyre, vha. en stabel. I det man leser et tall, legges det øverst på stabelen, mens i det man leser en operator, hentes – og fjernes – to øverste tall fra stabelen, anvendes operator på dem, og så legges resultatet øverst på stabelen. Eksemplene viser sekvenser av stabeler, der hver stabel stammer fra forrige i det tegnet som står under stabelen er lest.

(a)

		3				
	2	2	6		4	
11	11	11	11	17	17	13
11	2	3	$*$	$+$	4	$-$

(b)

	2		3		4	
11	11	22	22	25	25	21
11	2	$*$	3	$+$	4	$-$

(c)

			3			
		2	2	6		
	11	11	11	11	17	
4	4	4	4	4	4	-13
4	11	2	3	$*$	$+$	$-$

Programmer en aksjon `eval::IO()` (og alle hjelpefunksjoner) som leser fra terminalen én linje om gangen, med ett ikke-negativt heltall eller én operator, inntil brukeren taster en tom linje (dvs., kun Enter/CR knappen) for å avslutte programmet. Etter hver innlest linje, viser programmet stabelen resulterende fra evaluering av hele uttrykket som har blitt lest så langt, som i eksemplene over.
NB! Utskriften skal bestå kun av en liste med tall tilsvarende stabelen. F.eks., i tre eksemplene over, skriver programmet lister som følger, etter hver input fra brukeren (dvs., ett tall, +, * eller –, med Enter/CR knappen etter hver av dem):

(a)	(b)	(c)
`> eval`	`> eval`	`> eval`
11	11	4
[11]	[11]	[4]
2	2	11
[2,11]	[2,11]	[11,4]
3	*	2
[3,2,11]	[22]	[2,11,4]
*	3	3
[6,11]	[3,22]	[3,2,11,4]
+	+	*
[17]	[25]	[6,11,4]
4	4	+
[4,17]	[4,25]	[17,4]
–	–	–
[13]	[21]	[-13]

Når brukeren taster et ugyldig input (feil tegn, eller en operator når det ikke er nok argumenter), skal programmet gi en passende tilbakemelding og fortsette evaluering av uttrykket som har blitt samlet opp, etter at brukeren gir korrekt input.

4 Hindley-Milner (20%)

La fuksjonen `apply` være definert ved likningen: `apply f x = f x`.

4.1. Skriv denne definisjonen ved hjelp av λ-uttrykk, dvs. på formen `apply = \f ->...`

4.2. Bruk Hindley-Milner samt unifikasjonsalgoritme for å avlede typen til `apply`.

Lykke til!
Michał Walicki

Vedlegg

Hindley-Milner typeinferens: transformasjonsalgoritme

	input		output
(t1)	$E(\Gamma \mid con :: t)$	$\Rightarrow$	$\{t = \theta(con)\}$
			– typen til en konstant slås opp i ordboken
(t2)	$E(\Gamma \mid x :: t)$	$\Rightarrow$	$\{t = \Gamma(x)\}$
			– typen til en variabel sjekkes i konteksten
(t3)	$E(\Gamma \mid f\ g :: t)$	$\Rightarrow$	$E(\Gamma \mid g :: a) \cup E(\Gamma \mid f :: a \to t)$
			– a er en *fersk* typevariabel
(t4)	$E(\Gamma \mid \backslash x \to ex :: t)$	$\Rightarrow$	$\{t = a \to b\} \cup E(\Gamma, x :: a \mid ex :: b)$
			– a, b er *ferske* typevariabler

Martelli-Montanari unifikasjonsalgoritme

	input		output	forutsatt at :
$(u1)$	$E, t = t$	$\Rightarrow$	E	
$(u2)$	$E, f(t_1 ... t_n) = f(s_1 ... s_n)$	$\Rightarrow$	$E, t_1 = s_1, ..., t_n = s_n$	
$(u3)$	$E, f(t_1 ... t_n) = g(s_1 ... s_m)$	$\Rightarrow$	NO	$f \neq g$ eller $n \neq m$
$(u4)$	$E, f(t_1 ... t_n) = x$	$\Rightarrow$	$E, x = f(t_1 ... t_n)$	
$(u5)$	$E, x = t$	$\Rightarrow$	$E[x/t], x = t$	$x \notin Var(t)$
$(u6)$	$E, x = t$	$\Rightarrow$	NO	$x \in Var(t)$

Slutt

Oppgave 1 – løsningsforslag (25%)

1.1: (d) 2
1.2: (a) gir [0,1,1,1,1] 3
(b) gir [0, og programmet henger, siden `tail nats` ikke terminerer 3
(c) gir [0,1,2,3,4] 2
(d) gir [1,2,3,4,5] 2
1.3. (a) concat [”ab”,”cd”,””,”efg”] = [”ab”,”cd”,””,”efg”] – en kopi av inputlisten 2
(b) concat [”ab”,”cd”,””,”efg”] = ”abcdefg” – konkatenerer lister 2
(c) concat [”ab”,”cd”,””,”efg”] = ikke terminerer (mangler concat []=...) 3
(d) her `concat::[[a]]->[[a]->[a]]`, og

```
concat ["ab","cd","","efg"] = [(++)"ab",(++)"cd",(++)"",(++)"efg"]
```
3

– denne listen av funksjoner kan ikke vises, men kunne anvendes i en passende kontekst.

1.4. (b) `(a->b)->a->b` 2

hvis det er $\geq$ 23 poeng: +1

Oppgave 2 – løsningsforslag (20%)

Litt for enkelt...

```
-- r-te rad i matrisen ma
row ma r = ma!!(r-1)                                                    2
-- k-te kolonne i matrisen ma
col ma k = map (!!(k-1)) ma                                             2
-- listen med alle kolonner
cols ma = map (\x -> col ma x) [1..(length ma)]                         4
-- multiplikasjon av en enkel rad r og kolonne k
multrc r c = sum [x*y | (x,y) <- zip r c]                               4
-- hovedmetoden
mult ma mb = [map (multrc (row ma x)) (cols mb) | x <- [1..length ma]]  8
```

Oppgave 3 – løsningsforslag **(35%)**

listeargumentet `stab`: 5, IO(getLine, let):10, generelt oppsett:10, korrekthet:10.

```
eval = postio []
postio :: [Int] -> IO ()
postio stab = do
  ls <- getLine
  let inp = filter (/= ' ') ls
  if (inp =="") then return ()
  else if isDigit (head inp) then do
    let stabNxt = ((read inp)::Int):stab
    putStrLn (show stabNxt)
    postio stabNxt
  else if (not (elem (head inp) ['*','+','-'])) then do
     putStrLn ("Ugyldig tegn " ++ [head inp])
     postio stab
  else if (length stab < 2) then do putStrLn ("Mangler et argument.")
                                    postio stab
  else do
     let a = head stab
         b = head (tail stab)
         st = tail (tail stab)
         r = case (head inp) of
                   '*' -> a * b
                   '+' -> a + b
                   '-' -> b - a
     putStrLn (show (r:st))
     postio (r:st)
```

Oppgave 4 – løsningsforslag **(20%)**

4 **5.1.** `apply = \f -> \x -> f x`

16 **5.2.**

$\emptyset$	$\lambda f \to \lambda x \to f\ x :: t$	$\emptyset$
$f :: s$	$\lambda x \to f\ x :: b$	$t = s \to b$
$f :: s, x :: a$	$f\ x :: c$	$t = s \to b, b = a \to c$
$f :: s, x :: a$	$f :: d \to c$	$t = s \to b, b = a \to c$
$f :: s, x :: a$	$x :: d$	$t = s \to b, b = a \to c$
$unifiserer$		$t = s \to b, b = a \to c, s = d \to c, d = a$
		$t = s \to b, b = a \to c, s = \mathbf{a} \to c, d = a$
		$t = (a \to c) \to (a \to c), b = a \to c, s = a \to c, d = a$
$dermed$	$apply :: (a \to c) \to a \to c$	