Information

Skriftlig eksamen i IN3040
2020 Høst
Varighet: 07.12.2020 15:00 til 07.12.2020 18:00

Det er viktig at du leser denne forsiden nøye før du starter.

Generelt:

- Det er viktig at du sjekker emnets semesterside jevnlig. Eventuelle viktige beskjeder under eksamen blir gitt direkte fra faglærer på semestersiden.
- Husk at besvarelsen skal være anonym, du skal ikke oppgi navnet ditt.
- Alle hjelpemidler er tillatt ved skriftlig hjemmeeksamen. Det er opp til deg å innhente informasjon fra tilgjengelige kilder, vurdere kvaliteten og sette det hele sammen til en besvarelse basert på egen bearbeiding av stoffet. Besvarelsen som leveres skal reflektere eget kunnskapsnivå.

Samarbeid:

- Oppgaven skal løses individuelt. Samarbeid med andre anses som fusk.

Digital håndtegning:

- Om denne besvarelsen skal inneholde "digital håndtegning"/spesialtegn, bruk dine foretrukne verktøy til dette, f.eks. bilde av papirark fra mobil. Send bildet fra mobilen til deg selv på mail. Åpne bildet fra mailen på din "eksamensmaskin" og lagre bildet på denne maskinen. Herfra kan du laste opp fila til eksamensoppgaven. Det er viktig at tegningene er leselige. Husk å spesifisere hvilke tegninger som tilhører hvilke oppgaver. Det er studentens ansvar å dobbeltsjekke at bildefil er lastet opp og er leselig. MN anbefaler at du tar vare på originaltegninger til sensur har falt. Du kan også bruke nettbrett/elektronisk penn til å levere håndtegning. Om din eksamen inkluderer innlevering av digital håndtegning, har du fått 30 min. ekstratid.
- Sjekk MNs/UiOs anbefalte løsninger for bruk av digital håndtegning og spesialtegn

Kontaktinfo:

- Brukerstøtte eksamen

Lykke til!

Du står fritt til å løse oppgaven med SML eller Haskell. Vi gir oppgaveteksten her bare i SML-syntaks på grunn av begrensninger i Inspera-systemet.

1 SML: removeDuplicates

Definer en funksjon **removeDuplicates** som tar en liste av heltall *xs* og gir tilbake en liste *ys*, slik at *ys* inneholder hvert element fra *xs* bare en gang.

Eksempel:

- removeDuplicates [1,1,2,1,3,2,1,3];

val it = [1,2,3] : int list

Obs: du får 100% av mulige poeng for et riktig svar som bruker *higher-order functions* i dette spørsmålet, ellers 80% av poengene hvis du ikke bruker dette.

Skriv ditt svar her

```
1
```

Maks poeng: 5

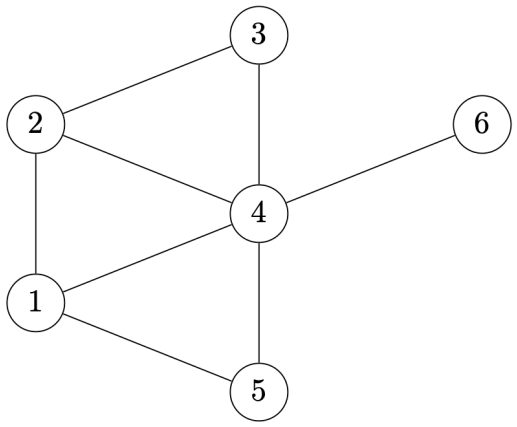
2 SML: Graph

Vi representerer en graf som en liste av kanter ved hjelp av data-typen **Graph**:

```
datatype Graph = Edges of (int * int) list;
```

Kanter er par av heltall-verdier (en verdi gir et unikt navn til hver node). Tegningen nedenfor viser grafen gitt gjennom:

```
val g = Edges [(1,2),(1,4),(1,5),(2,3),(2,4),(3,4),(4,5),(4,6)];
```



Du kan anta følgende funksjon som gitt i dine løsninger:

```
(* val flatten = fn : ('a * 'a) list -> 'a list *)
fun flatten [] = []
  | flatten ((x,y)::zs) = (x::y::flatten zs);
```

a) Definer en funksjon **nodes = fn : Graph -> int list** som leverer listen av alle noder i grafen. Enhver node skal forekomme bare en gang i resultat.

Eksempel:

```
- nodes g;
val it = [1,2,4,5,3,6] : int list
```

Obs: du får 100% av mulige poeng for et riktig svar som bruker *higher-order functions* i dette spørsmålet, ellers 80% av poengene hvis du ikke bruker dette.

b) Definer en funksjon **isElem = fn : int -> Graph -> bool** som gir som resultat sant hvis og bare hvis, gitt en node *n* og grafen *g*, *n* er i listen av **nodes g**.

Obs: du får 100% av mulige poeng for et riktig svar som bruker *higher-order functions* i dette spørsmålet, ellers 80% av poengene hvis du ikke bruker dette

c) En *skog* er en graf uten løkker, og det finnes på det meste én sti som rekker frem til enhver node.

Tegningen øverst viser derfor *ingen* skog.

Definer en funksjon **spanningForest = fn : Graph -> Graph** som gitt en graf *g* beregner en spennskog *w*, slik at:

- **nodes g = nodes w**,
- *w* er en skog,
- enhver kant av *w* er også et element i g.

Den tomme graf er en spennskog. Ved å legge til en kant *(m,n)* i en skog som inneholder *hverken m* heller *n* gir igjen en skog. Du kan bruke funksjonen **isElem** fra før selv om du har ikke løst del-spørsmål (b).

Skriv ditt svar her

Maks poeng: 10

Oversette den rekursive funksjonen til en hale-rekursiv definisjon. Du kan anta at funksjonen f er kommutativ og assosiativ. Vær obs på at alle hjelper-funksjoner du bruker også må være hale-rekursive!

Spørsmål kan besvares med SML eller Haskell.

3 Tail recursion (C-I)

SML:

```
datatype 'a r = C of 'a | I of ('a r * 'a r);
fun calc f i (C a) = f i a
  | calc f i (I (x,y)) = f (calc f i x) (calc f i y);
```

Fill in your answer here

```
1
```

Haskell:

```
data R a = C a | I (R a , R a)
calc f i (C a) = f i a
calc f i (I (x,y)) = f (calc f i x) (calc f i y)
```

Maks poeng: 5

Beregn typen til det følgende uttrykket i henhold til MLs algoritme for typeinferens:

1. Tegn parseringsgraf ("parse graph")
2. Bruk type-variabler for alle noder hvor dette er nødvendig. Ta hensyn til typer av konstanter og funksjoner gitt nedenfor.
3. Utled de tilhørende ligningene og løs det resulterende ligningssettet.

Vis mellom-trinnene i utregninger, dvs. hvordan du løser ligningene. Angi om typeinferens oppdager en typefeil, og hvordan.

Antar som gitt funksjonen:
`ite = fn : bool → a → a → a`

4 ITE-1

`fn f ⇒ fn b ⇒ ite (f b) (f 42) (f 3040)`

Upload your file here. Maximum one file.

Alle filtyper er tillatt. Maksimal filstørrelse er **2 GB**.

Velg fil for opplasting

Maks poeng: 10

Modeller de gitte objektorienterte datastrukturene i SML eller Haskell.
Implementer operasjonen **value = fn : Course -> int** og operasjonen **double()** for de typene der det er nødvendig. Merk at spørsmålet er delt inn i delene (a) & (b). På grunn av begrensninger i Inspera, vennligst skriv inn begge svarene i det samme svarfeltet. Skill dem f.eks. med noen blanke linjer eller kommentarer.

5 SML-DS-2

a)

```
interface Course {
  int value();
}
class IN3040 implements Course {
  int value() { return 3040; }
}
class IN9040 implements Course {
  int i;
  int value() { return i; }
}
```

b)

```
interface Course {
  int value();
}
class IN3040 implements Course {
  int value() { return 3040; }
}
class IN9040 implements Course {
  int i;
  int value() { return i; }
  (* New: *)
  int double(Course o) { return o.value() * value(); }
}
```

I del (b) trenger du *ikke* å modellere at OO-kode kan ha en null-pointer exception.

Skriv ditt svar her

```
1
```

Maks poeng: 10

6 FO-1

Angi hvilket svar som fanger betydningen av Prolog predikatet **`p/1`**:

p(X) :- q(X,Y).
p(X) :- q(Y,X).

Velg ett alternativ:

- p(X) er sant, hvis og bare hvis begge q(X,Y) og q(Y,X) er sant for noen Y.
- p(X) er sant, hvis og bare hvis minst en av q(X,Y) eller q(Y,X) er sant for noen Y.
- For alle X og for alle Y, p(X) og q(X,Y) og q(Y,X) er sant.

Maks poeng: 3

7 PL-DS-Lists-1

Vi kan definerer medianen til en sortert liste med et ulikt antall elementer på følgende måte:

- hvis listen inneholder bare ett eneste element, er dette medianen.
- ellers finner vi medianen rekursivt ved å ta bort første og siste element i listen.

For eksempel, medianen av listen 3, 5, 7, 12, 13, 14, 21, ***23***, 23, 23, 23, 29, 39, 40, 56 er 23.

a) Definer et Prolog-predikat **median/2** slik at **median(XS,Y)** er sant hvis **Y** er medianen av **XS**.
b) Beskriv med dine egne ord hva spørsmålet **?- median(XS,42)** betyr og hva Prolog-systemet ville svare.

Skriv ditt svar her

```
1
```

Maks poeng: 7

8 PL-DS-Heap

Vi definerer en datastruktur for trær som lagrer verdier i nodene på følgende måte:

empty indikerer det tomme tre. **node(V,L,R)** indikerer et tre med verdi V i noden, og venstre og høyre sub-trær L,R.

a) Definer et predikat **depth(T,N)** som er sant hvis treet T har dybde (depth) N. Det tomme tre (empty) har dybde 0, og en node har dybde 1+M, hvor M er maksimum av dybdene for de to sub-trærne.

b) Definer et predikat **heap(H,N)**, som er sant hvis treet H har formen til en 'heap' med dybde N:

- Det tomme (**empty**) tre er en *heap* med dybde 0.
- En node **node(V,L,R)** er en *heap*, hvis
  - L og R er *heaps*, og
  - hvis L og/eller R ikke er tomme (**empty**), så er verdien på deres topp mindre enn eller lik V, og
  - forskjellen i dybdene av L og R er på det meste 1.

Bemerk at man ved passelig bruk av det andre argumentet til heap ikke egentlig trenger å bruke predikatet som er definert i del-spørsmål (a)!

Skriv ditt svar her

```
1
```

Maks poeng: 10

9 Static vs dynamic typing

Det finnes en stor mengde programmeringsspråk, og disse er forskjellige på mange måter. En viktig forskjell på språk er hvordan typesystemet/typesjekkingen fungerer. Vanligvis skiller vi mellom statisk typede og dynamisk typede/sjekkede språk. Forklar kort hva forskjellen på disse er, og hvilke fordeler og eventuelle ulemper begge varianter kan gi.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

10 Static typing - conservative

Hva mener vi når vi sier at statisk typede programmeringsspråk er «konservative», og bare kan finne en nedre skranke («lower bound») for hvor korrekt programmet er? Hvordan skiller dette seg fra dynamisk typede språk?

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

11 Runtime stack 1

Programmeringsspråket C# er et (stort sett) statisk typet språk, som ligner ganske så mye på Java, både når det gjelder syntaks og semantikk.

Det har i tillegg typeinferens (når man bruker nøkkelordet «var»), og en del konstruksjoner inspirert av funksjonelle språk. Superklaser for en subklasse angis etter et kolon («:») i stedet for «extends» som i Java. Programmets eksekvering starter med metoden Main.

Gitt følgende program:

```
public class MyProgram
{
    public static void Main(String[] args)
    {
        var prog = new MyProgram();
        prog.DoTheThings();
    }

    public void DoTheThings()
    {
        // Gaute and Eyvind are students:
        var eyvind = new Student { Age = 41, Name = "Eyvind" };
        var gaute = new Student {
            Age = 25 /* don't really know, but he looks young and fresh! */,
            Name = "Gaute"
        };

        // Fride (Eyvind's wife) is not a student, she is just a nice person:
        var fride = new Person { Age = 41, Name = "Fride" };

        var people = new List<Person> { eyvind, gaute, fride };
        var youngAndPromising = FilterTheYoung(people);
        youngAndPromising.ForEach(p ⇒ Console.WriteLine(p.Name));
    }

    public List<Person> FilterTheYoung(List<Person> people)
    {
        var filteredList = new List<Person>();
        foreach(var p in people)
        {
            if(p.Age ⩽ 40)
                filteredList.Add(p);
        }
        // HERE
        return filteredList;
    }
}

public class Person {
    public int Age {get; set; }
    public string Name {get; set; }
}

public class Student : Person {
    public String FavoriteSubject { get; set; } = "IN3040"; // obviously, duh
}
```

Tegn kjøretidsstakken ("the runtime stack"), inkludert objekter, når eksekveringen har nådd punktet merket "HERE", altså rett før den filtrerte listen returneres.

Angi eventuelle forutsetninger du gjør deg eksplisitt.

Last opp filen her. Maks én fil.

Alle filtyper er tillatt. Maksimal filstørrelse er **2 GB**.

Velg fil for opplasting

Maks poeng: 7

12 Variance

Vi ser igjen på programmet fra forrige spørsmål:

```
public class MyProgram
{
    public static void Main(String[] args)
    {
        var prog = new MyProgram();
        prog.DoTheThings();
    }

    public void DoTheThings()
    {
        // Gaute and Eyvind are students:
        var eyvind = new Student { Age = 41, Name = "Eyvind" };
        var gaute = new Student {
            Age = 25 /* don't really know, but he looks young and fresh! */,
            Name = "Gaute"
        };

        // Fride (Eyvind's wife) is not a student, she is just a nice person:
        var fride = new Person { Age = 41, Name = "Fride" };

        var people = new List<Person> { eyvind, gaute, fride };
        var youngAndPromising = FilterTheYoung(people);
        youngAndPromising.ForEach(p ⇒ Console.WriteLine(p.Name));
    }

    public List<Person> FilterTheYoung(List<Person> people)
    {
        var filteredList = new List<Person>();
        foreach(var p in people)
        {
            if(p.Age ⩽ 40)
                filteredList.Add(p);
        }
        // HERE
        return filteredList;
    }
}

public class Person {
    public int Age {get; set; }
    public string Name {get; set; }
}

public class Student : Person {
    public String FavoriteSubject { get; set; } = "IN3040"; // obviously, duh
}
```

Er det at vi kaller på metoden FilterTheYoung med en liste som inneholder både Person og Student-objekter trygt og typesikkert?

Begrunn svaret ditt.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

13 Stack with function values

Vi ønsker å generalisere filtreringsmetoden vi brukte i de forrige oppgavene. Vi har derfor tatt i bruk C# sine muligheter for funksjonsparametre, og skrevet en generell Filter<T>-funksjon, som vi benytter til å filtrere ut de samme unge personene fra lista som tidligere.

I C# deklareres funksjonsparametre ved hjelp av en generisk type Func<T1..TN>, der siste typeparameter angir returtypen:

```
public void DoTheThings()
{
    // Gaute and Eyvind are students:
    var eyvind = new Student { Age = 41, Name = "Eyvind" };
    var gaute = new Student {
        Age = 25 /* don't really know, but he looks young and fresh! */,
        Name = "Gaute"
    };

    // Fride (Eyvind's wife) is not a student, she is just a nice person:
    var fride = new Person { Age = 41, Name = "Fride" };

    var people = new List<Person> { eyvind, gaute, fride };
    var youngAndPromising = Filter(AgeFilter, people);

    youngAndPromising.ForEach(p ⇒ Console.WriteLine(p.Name));
}

bool AgeFilter(Person p) {
    return p.Age ⩽ 40;
}

public List<T> Filter<T>(Func<T, bool> predicate, List<T> items) {
    var list = new List<T>();
    foreach(var i in items)
    {
        if(predicate(i))
            list.Add(i);
    }
    return list;
}
```

Anta at resten av programmet er som i de foregående oppgavene.

Tegn kjøretidsstakken ("the runtime stack") til dette programmet rett før return-setningen i funksjonen AgeFilter utføres.

Last opp filen her. Maks én fil.

Alle filtyper er tillatt. Maksimal filstørrelse er **2 GB**.

Velg fil for opplasting

Maks poeng: 8

14 Dynamic

I C# så kam programmereren gjøre deler av programmet dynamisk typet. Nedenfor ser vi et eksempel på dette, med variabelen student (understreket med hvitt):

```
public void DoTheThings()
{
    // Gaute and Eyvind are students:
    var eyvind = new Student { Age = 41, Name = "Eyvind" };
    var gaute = new Student {
        Age = 25 /* don't really know, but he looks young and fresh! */,
        Name = "Gaute"
    };

    // Fride (Eyvind's wife) is not a student, she is just a nice person:
    var fride = new Person { Age = 41, Name = "Fride" };

    var people = new List<Person> { eyvind, gaute, fride };
    var youngAndPromising = Filter(AgeFilter, people);

    youngAndPromising.ForEach(p ⇒
    {
        dynamic student = p;
        Console.WriteLine(student.FavoriteSubject);
    });
}
```

Denne linjen gjør alle oppslag fra variabelen student dynamiske, hvilket innebærer at de vil bli sjekket under kjøring, og ikke under kompilering. (Resten av programmet er som før.)

Hvordan påvirker dette eksekveringen av programmet, og spesielt utskriften av student.FavoriteSubject? Ville programmet ha vært gyldig uten at vi tok i bruk dynamisk typing på dette punktet? Sier dette i så fall noe generelt om statiske vs dynamiske typesystemer?

Begrunn svaret ditt.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

15 Dynamic scope

Som vi så i det forrige spørsmålet, så har programmereren i C# mulighet for å ta i bruk dynamisk typing der det er nyttig.

Språket har dog i all hovedsak statisk *skop* (med unntak for exceptions, som vi har snakket om på forelesning).

Dersom du skulle innført valgfritt bruk av dynamisk skop i C#, hvordan ville du gjort det? Hvordan kunne dette sett ut (syntaktisk)? Hvilke utfordringer vil dette føre til for språkdesigner og/eller programmerer?

Forklar kort og uformelt.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5