Forside

Skriftlig hjemmeeksamen i IN3040
2021 Høst

Varighet: 10. desember 09:00 til 10. desember 13:30.

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

Generell informasjon:

- Viktige beskjeder under eksamen blir gitt direkte fra faglærer på emnets semesterside eller i Canvas, om emnet ditt bruker det. Det er derfor viktig at du sjekker emnets semesterside/Canvas-rommet jevnlig.
- Besvarelsen din skal reflektere ditt eget, selvstendige arbeid og skal være et resultat av din egen læring og arbeidsinnsats.
- Alle hjelpemidler er tillatt ved skriftlig hjemmeeksamen. Dersom du gjengir tekst fra bøker, nettartikler eller lignende, så må det henvises til disse kildene i besvarelsen for å unngå mistanke om ulovlig tekstlikhet. Dette gjelder også dersom du oversetter tekst fra andre språk.
- Du er selv ansvarlig for å sørge for at eksamensbesvarelsen din ikke er tilgjengelig for andre under eksamenstiden, hverken fysisk eller digitalt.
- Husk at besvarelsen skal være anonym, du skal ikke oppgi hverken ditt eller medstudenters navn.
- Om du vil trekke deg fra eksamen, trykk på hamburgermeny oppe til høyre i Inspera og velg "Jeg vil trekke meg".

Emner på denne eksamenen:
På samme måte som forelesningene og pensum har vært oppdelt i tematiske bolker, så er også eksamen strukturert i tre deler, med tilhørende andel av oppnåelige poeng:

- Prolog 20 %
- SML 40 %
- Objektorientering, kjøretid, utvidbarhet, etc. 40 %

Spørsmål under eksamen:
Dersom du har spørsmål til oppgaven, så kan du sende epost til faglærere:

- Volker Stolz: stolz@ifi.uio.no
- Eyvind W. Axelsen: eyvindwa@gmail.com

Samarbeid under eksamen:
Det er ikke tillatt å samarbeide eller kommunisere med andre under eksamen. Samarbeid og kommunikasjon vil bli betraktet som forsøk på fusk. Det blir gjort plagiatkontroll av alle innleverte eksamener der tekstlikhet mellom besvarelser blir sjekket. Om du bruker notater som er utarbeidet i samarbeid med andre før eksamen, kan dette gi treff i en plagiatsjekk. Slik tekstlikhet kan bli betraktet som lav selvstendighet eller forsøk på fusk. Unngå derfor klipp/lim fra notater laget i samarbeid med andre.

Fusk:
Les om hva som regnes som fusk på UiOs nettsider.

Digital håndtegning/filopplasting:
Du har fått 30 min. ekstratid til opplasting av filer (f.eks. digital håndtegning). Sjekk hvordan du leverer inn digital håndtegning

Kontaktinfo:
Brukerstøtte eksamen

1 H21-PL-even-1

Les følgende definisjon nøye:

even(0).
even(N) :- X is N-2, even(X).

Forklar Prolog-systemets oppførsel hvis vi spør etter *alle* løsninger til følgende forespørsler:

1) **?- even(your_candidate_id).**
2) **?- even(Z).**

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

2 H21-PL-Func-1

Anta faktaene nedenfor. Skriv predikatet **avg/2** som er sant hvis **I** i spørringen **avg(C,I)** er det gjennomsnittlige antallet innbyggere i landet **C**. Bruk Prologs aritmetikk med **"is"** for å beregne verdien.

inhab(usa,329).
inhab(germany,83).
inhab(denmark,5).

area(denmark,42933).
area(germany,357386).
area(usa,9834000).

Skriv ditt svar her

```
1
```

Maks poeng: 10

3 H21-SML-T-3

Vennligst definer en funksjon (med eventuelle ekstra definisjoner der det er nødvendig) som har følgende type signatur:

SML:
val f = fn : 'a → ('a → 'a) list

Haskell:
f: a → [a → a]
Skriv ditt svar her

```
1
```

Maks poeng: 5

4 H21-SML-T-2

Vennligst definer en funksjon (med eventuelle ekstra definisjoner der det er nødvendig) som har følgende type signatur:

SML:
val f = fn : 'a → ('a → 'a) option

Haskell:
f: a → Maybe (a → a)
Skriv ditt svar her

```
1
```

Maks poeng: 5

5 H21-SML-T-1

Vennligst definer en funksjon (med eventuelle ekstra definisjoner der det er nødvendig) som har følgende type signatur:

SML:
val f = fn : 'a → 'a → 'b

Haskell:
f: a → a → b
Skriv ditt svar her

```
1
```

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.

6 H21-TI-1

fn x => fn y => (x(y)) (y)



Last opp din fil her. Maksimum en fil.

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

Velg fil for opplasting

Maks poeng: 10

Gitt følgende SML definisjoner:

```
exception E of int;

fun f1 (g : int * int -> int) (a : int) (nil : int list) : int = if a = 0 then raise E(0) else a
|    f1 g a (y::ys) = f1 g (g (y,1)) ys handle E(e) => a+e;

fun f2 (g : int * int -> int) (a : int) (nil : int list) : int = if a = 0 then raise E(0) else a
|    f2 g a (y::ys) = f2 g (g (y,1)) ys;
```

7 H21-Tailrec-Q1

Kryss av for hvilke funksjoner som er hale-rekursive!
Velg ett eller flere alternativer

- [ ] f1
- [ ] f2
- [ ] hverken f1 heller f2

Maks poeng: 5

8 H21-Tailrec-Q2

Gjør rede for hvorfor **f1** bruker mye mer av stacken enn **f2** i lys av ditt svar ovenfor!

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

Given are the following datatype definitions:

```
datatype T = T of int
and U = UA of T | UB of T * T;

datatype V = A of int | B of V * V;

datatype W = C of int * int;

datatype X = D of int * X | E of int * X;

datatype Y = F of int | G of int * Y;

datatype Z = H | I of int * Z;
```

9 H21-SML-Datatypes-1

Which of the datatypes could be used to represent a pair of integers? In other words: in which data structures can we construct an expression that contains at least two integers?

Velg ett eller flere alternativer

- [ ] U
- [ ] V
- [ ] W
- [ ] X
- [ ] Y
- [ ] Z

Maks poeng: 5

10 H21-SML-Datatypes-2

Which of the datatypes are *recursive*?
Velg ett eller flere alternativer

- [ ] U
- [ ] V
- [ ] W
- [ ] X
- [ ] Y
- [ ] Z

Maks poeng: 5

11 H21-SML-HO

Gitt følgende definisjoner:

```
datatype Suit = Hearts | Clubs | Diamonds | Spades;
datatype Rank = Val of int | Jack | Queen | King | Ace;
datatype Card = Card of Rank * Suit;
fun isRed Hearts = true
  | isRed Diamonds = true
  | isRed _ = false;
fun isDiamond Diamonds = true
  | isDiamond _ = false;
fun isAce Ace = true
  | isAce _ =  false;
fun isLow (Val i) = i < 5
  | isLow _ = false;

fun lowDiamond nil = nil
  |   lowDiamond (Card(r,s)::cs) = (isLow r andalso isDiamond s) :: lowDiamond cs;
fun redAces nil = nil
  |   redAces (Card(r,s)::cs) = (isAce r andalso isRed s) :: redAces cs;
fun lowRed nil = nil
  |   lowRed (Card(r,s)::cs) = (isLow r andalso isRed s) :: lowRed cs;
```

Skriv en funksjon
val selectCards = fn : (Rank -> bool) -> (Suit -> bool) -> Card list -> Card list
og erstatt kroppene til **lowDiamond, redAces** og **lowRed** med et kall til den nye funksjonen.
Du kan også bruke definisjoner fra SML sin **List** bibliotek.
Skriv ditt svar her

```
1
```

Maks poeng: 5

12 A: OO hierarchies

Spørsmål A:

Gitt følgende Java-kode

```
public class Zoo {
  Animal[] animals = populateZoo(); // assume this method fills the zoo with animals

  public static void main(String[] args) {
    Zoo zoo = new Zoo();
    zoo.printAnimals();
  }

  public void printAnimals() {
    for(Animal a : animals) {
      System.out.print(a.print() + ". ");
    }
  }
}

abstract class Animal { abstract String print(); }
class Lion extends Animal { public String print() { return "I'm a lion"; } }
class Pig extends Animal { public String print() { return "I'm a pig"; } }
```

Inne i metoden **printAnimals()**, så er det et kall til **a.print()**, hvor intensjonen er å kalle den riktige print-metoden avhengig av hva slags **Animal** variabelen **a** peker på. Vil dette fungere som ønsket? Hvis ja, forklar hva det er som får det til å fungere. Hvis nei, forklar hvorfor det ikke fungerer. Bruk terminologi fra forelesningene i din forklaring.

Skriv ditt svar her

Words: 0

Maks poeng: 6

13 B: Runtime stack

Spørsmål B:

Tegn kjøretidsstacken ("the runtime stack") for programmet fra spørsmål A, på det punktet hvor en **print()**-metode kalles i en subklasse av **Animal**.

Dersom du svarte "nei" i spørsmål A, eller ikke svarte i det hele tatt, kan du nå anta at programmet fungerer etter intensjonen.

Dersom du gjør noen egne forutsetninger ut over dette i din tolkning, så skriv disse eksplisitt inn i tegningen.

Last opp filen her. Maks én fil.

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

Velg fil for opplasting

Maks poeng: 8

14 C: OO extensibility

Spørsmål C:

Vi ønsker nå å utvide vårt zoo-program fra spørsmål A, slik at det kan ha flere dyr, og vi kan intereagere litt mer med disse dyrene. Helt spesifikt så ønsker vi oss:

- En ny type dyr, ape, representert ved en ny **Monkey**-klasse
- At det skal være mulig å mate alle dyrene, ved å bruke en metode **feed()**. Dette skal skrives på en slik måte at vi i klassen **Zoo** kan lage metoden **feedAnimals()** etter samme lest som **printAnimals()** ble skrevet, slik at vi inne i en løkke kan kalle hvert dyr (**Animal)** sin **feed()**-metode.

Men, vi har et problem. Kildekoden for **Animal**, **Lion** og **Pig**-klassene fra spørsmål A ble skrevet av den strenge og monopolistiske programvarebedriften ZooSoft Inc., og deres lisens gjør at det ikke er tillatt for oss å endre på den originale kildekoden.

Dette gjør det litt vanskelig for oss å få til alt vi ønsker, det vil si, begge de to punktene listet opp ovenfor. Forklar hva det er vi relativt enkelt kan få til, og hva som ikke er så lett, og hvorfor det er slik. Tenk i kontekst av typisk objektorientert programmering, og hva man typisk kan få til med dette.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 6

15 D: FP extensibility

Spørsmål D:

På same måte som i forrige spørsmål, så ønsker vi nå å utvide vår zoo slik at det kan ha flere dyr, og vi kan intereagere litt mer med disse dyrene. Helt spesifikt så ønsker vi oss:

- En ny type dyr, ape, representert ved en ny **Monkey**-datatype
- At det skal være mulig å mate alle dyrene, ved å bruke en funksjon **feed**. Dette skal skrives på en slik måte at kan lage funksjonen **feedAnimals** etter samme lest som **printAnimals** ble skrevet, slik at vi for hvert dyr kan kalle **feed**-funksjonen.

Nok en gang så er koden skrevet av strenge, monopolistiske, ZooSoft Inc., og lisensen lar oss ikke endre noe av koden under. Men, til forskjell fra forrige spørsmål, så er den opprinnelige koden nå skrivet i SML:

```
datatype Animal = Pig | Lion

val animals = (* a list of animals from somewhere *)

fun print a =
  case a of Pig => "I'm a pig"
      |     Lion => "I'm a lion"

fun printAnimals (a :: rest) = print a ^ ". " ^ printAnimals(rest)
  | printAnimals nil = ""
```

Får vi det samme problemet her som i spørsmål C? Eller er det noe som er grunnleggende annerledes i funksjonelle språk. Forklar under.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 4

16 E: Runtime stack with first class functions

Spørsmål E:

Når vi hadde **Zoo**-klassen fra spørsmål A, så var det litt trist at **printAnimals**-metoden ikke printet noe særlig spennende greier. Vi ønsker derfor å gjøre den mer fleksibel, ved å endre den slik at den tar inn en funksjonsparameter som tar seg av selve printing til konsollet. Koden blir da som følger, i Java:

```
interface AnimalPrinter {
  void printAnimal(Animal a);  // AnimalPrinter is a "functional interface"
}

public class Zoo2 {
  Animal[] animals = populateZoo(); // assume this method fills the zoo with animals

  public static void main(String[] args) {
    Zoo2 zoo = new Zoo2();
    AnimalPrinter fancyPrint = a -> System.out.print("Look! " + a.print() + ". ");
    zoo.printAnimals(fancyPrint);
  }

  public void printAnimals(AnimalPrinter ap) {
    for(Animal a : animals) {
      ap.printAnimal(a);
    }
  }
}

abstract class Animal { abstract String print(); }
class Lion extends Animal { public String print() { return "I'm a lion"; } }
class Pig extends Animal { public String print() { return "I'm a pig"; } }
```

Tegn kjøretidsstacken ("the runtime stack") rett før eksekveringen returnerer fra et kall på **print()**-metoden i et **Animal**-objekt.

Last opp filen her. Maks én fil.

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

Velg fil for opplasting

Maks poeng: 8

17 F: Retroactive fanciness

Spørsmål F:

I mange situasjoner, så kan vi ønske oss å legge til nye "features" til eksisterende typer, uten å endre den opprinnelige kildekoden. I mange eksisterende språk, som f.eks. Java, så kan dette være litt problematisk.

I dette spørsmålet vil vi anta at det finnes et nytt språk, *Cava* (som på mange måter ligner på Java), som har en mekanisme kalt **retrofit**, som kan brukes til å *retroaktivt* legge til nye metoder, eller bytte ut eksisterende metoder, i en eksisterende klasse, uten at man må endre den opprinnelige kildekoden. Som et eksempel så antar vi at det følgende programmet er et gyldig Cava-program:

```
// in library1, that we cannot change the source code of, due to license constraints:
class A {
   void m() { ... implementation here ... }
   A a() { return new A(); }
}

// in library2, that we write from scratch ourselves:
import library1:
retrofit A with {
   replace void m() { ... new implementation here ... }
   add void x() { ... implementation here ... }
}
class B {
   public B() {
     A a = new A();
     a.a().m();
     a.a().x();
   }
}
```

1) Skriv en enkel EBNF-grammatikk for retrofit-mekanismen. Den bør minst støtte eksempelet vist over. Du kan gå ut fra at det finnes eksisterende definisjoner for identifikatorer, klasse- og metodedeklarasjoner og -kropper, etc.

2) Det er mange utfordringer med å skulle implementere en slik mekanisme på en trygg måte. Diskuter uformelt noen (maks 3) problemstillinger som du tenker det er viktig å ta med i betraktningen, og, om mulig, skisser mulige løsninger for disse, eller diskuter hvorfor de er vanskelige å løse på en god måte.

Skriv ditt svar her

Words: 0

Maks poeng: 8