MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Kursledare och examinator: Anders Mörtberg

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2023-03-14

- **Del 1** består av 8 flervalsfrågor där minst ett svarsalternativ är korrekt. Om man svarar fel eller inte har exakt rätt antal alternativ får man 0 poäng på frågan.
- **Del 2** består av ett antal frågor med varierande antal poäng vilka ska lösas genom att man skriver kod i de olika programmeringsspråken i kursen.
- **Skriv tydligt.** Svårlästa svar riskerar 0 poäng.
- Inga externa bibliotek får användas om det inte står explicit i uppgiften.
- Skriv bara på en sida av varje papper.
- För att få godkänt måste man ha minst 4 poäng på Del 1, har man inte det rättas inte Del 2.
- **Hjälpmedel:** Ett A4 med så mycket information du vill. Du får skriva på båda sidorna.
- **Betygsgränser:** E: 15, D: 18, C: 21, B: 24, A: 27, av maximala 30.

Del 1: flervalsfrågor (1p per fråga, 8p totalt)

Var snäll och samla svaren på del 1 på ett svarspapper.

1. Vilket/Vilka påståenden nedan är sanna?

A. Typklasser i Haskell är samma sak som klasser i Java.
B. Typfel hittas vid kompilering för statiskt typade språk.
C. En polymorf funktion är en funktion som tar in en funktion som argument.
D. Backtracking är en viktig del i logikprogrammering.
E. Man kan returnera arrayer från funktioner i C.

2. Hur skapar man en funktion `foo` som tar ett heltal `x` och ett flyttal `y` och returnerar en sträng i JavaScript?

A. `def foo(x,y):`
B. `function foo(x,y)`
C. `function foo(var x, var y)`
D. `string function foo(int x,float y)`
E. `string foo(int x,float y)`

3. Vad kommer skrivas ut om man kör C-koden till höger?

```
int x[3] = {1,2,3};
int *p = &x[0];
printf("%d", *(p+3));
```

A. `1`
B. `2`
C. `3`
D. `4`
E. Det går inte att säga då värdet som hämtas ut är på en minnesadress utanför arrayen `x`.

4. Vilken/Vilka Prolog kodsnuttar fångar påståendet att ”Everyone that eats pizza is happy”?

A. `happy(X) :- eats(X,pizza).`
B. `eats(X,pizza) :- happy(X).`
C. `eats(X,pizza), happy(X).`
D. `eats(X,pizza) :- X is happy.`
E. `:- if eats(X,pizza) then X is happy.`

5. Vad gäller för Java koden till höger?

A. `name` är ett `Person` objekt.

B. `nm` är ett instansattribut.

C. Koden är fel då man glömt att implementera `print_name` metoden.

D. Man kan *inte* skapa objekt som är instanser av `Person` klassen då den är abstrakt.

E. Koden är fel då Java inte stödjer abstrakta klasser.

```
abstract class Person {

    private String name;

    public Person(String nm) {
        this.name = nm;
    }

    public abstract void print_name();

}
```

6. Vilken/Vilka av följande uttryck motsvarar en del i en typisk implementation av ett statiskt typat programmeringsspråk?

A. Sorter

B. Lexer

C. Classifier

D. Reader

E. Typechecker

7. Vad är typen på Haskellfunktionen:

```
f x y z = map x y ++ snd z
```

A. `(a -> b) -> [a] -> (c,[b]) -> [b]`

B. `(a -> b) -> [a] -> ([c],[b]) -> [b]`

C. `[a] -> [b] -> ([b],[c]) -> [(a,b)]`

D. `(a -> [b]) -> [a] -> (c,[b]) -> [[b]]`

E. Ingen då koden leder till ett typfel.

8. Vilket/Vilka av följande påståenden stämmer för JavaScript?

A. JavaScript är statiskt typat.

B. Det finns pekare i JavaScript.

C. JavaScript är dynamiskt typat.

D. Kod skriven i JavaScript kan köras av en webbläsare.

E. Variabler i JavaScript kan byta typ efter att man deklararerat dem.

Del 2: kodfrågor (22p totalt)

Var snäll använd ett papper till varje uppgift i del 2. Lösningarna på deluppgifterna för varje programmeringsspråk kan skrivas på samma papper.

9. **Imperativ programmering i C**

Den här uppgiften går ut på att skriva en funktion

```
int sum_numbers(int* p, int n)
```

som tar in en `int`-pekare `p` och ett positivt tal `n`. Funktionen ska sedan returnera summan av talet som `p` pekar på och de `n` efterföljande talen i minnet.

Exempelanvändning:

Om man kör följande kodsnutt:

```
int main() {

  int x[5] = {1,2,3,4,5};
  int *p = &x[0];

  printf("%d\n", sum_numbers(p,5));
  printf("%d\n", sum_numbers(p,2));
}
```

så ska resultaten som skrivs ut vara `15` och `3`.

(a) Skriv funktionen med en `for`-loop. (3p)

(b) Skriv om funktionen så att den använder sig av `goto` istället för `for`-loopen. (2p)

10. Objektorienterad programmering i Java

Den här uppgiften går ut på att implementera klasser för att representera fordon.

(a) Skapa en klass `Vehicle` med de *publika* instansattributen `manufacturer` och `number_of_wheels` av typ `String` och `int`. Klassen ska även ha ett *privat* instansattribut `owner` av typ `String`. Utöver dessa ska klassen ha en konstruktor som låter användaren sätta de olika attributen samt en ”getter” metod för att hämta ut vem som äger fordonet (dvs `owner`). (2p)

(b) Skapa en klass `Car` som ärver `Vehicle`. Bilklassen ska ha det publika instansattributet `km` av typ `int` som representerar hur många kilometer bilen gått. Konstruktorn för bilar ska använda sig av konstruktorn för `Vehicle`, men sätta `number_of_wheels` till 4. Användaren ska alltså bara behöva ange tillverkare, ägare och hur många kilometer bilen gått när man anropar konstruktorn för bilar. (2p)

(c) Visa hur man skapar bilen `new_tesla` som är tillverkad av Tesla, ägd av Elon Musk och som har gått 0 kilometer. (1p)

11. Funktionell programmering i Haskell

(a) Skriv en *rekursiv* funktion `partition :: (a -> Bool) -> [a] -> ([a],[a])` som tar in en funktion `p` som returnerar `True` eller `False`, samt en lista `xs`. Funktionen ska sedan returnera ett par bestående av en lista med de element i `xs` som gör `p` sann och en lista med de element som gör `p` falsk. (3p)

Exempelanvändning:

```
> partition (\x -> x > 3) ([5,8,4,3,10,20])
([5,8,4,10,20],[3])
> partition (\x -> x > 3) ([2,5,8,4,3,10,20])
([5,8,4,10,20],[2,3])
```

Obs: för poäng får man *inte* använda den inbyggda Haskellfunktionen `filter`.

(b) Använd `partition` för att implementera Haskellfunktionen `all :: (a -> Bool) -> [a] -> Bool` som fungerar så att `all p xs` returnerar `True` om alla element i `xs` uppfyller `p` och `False` annars. (3p)

Obs: för poäng måste lösningen använda sig av `partition`, men man får använda `partition` även om man inte löst uppgift 11(a).

12. Logikprogrammering i Prolog

(a) På tentan 2022-03-14 skulle man implementera en Haskellfunktion för att beräkna Tribonacci talen. Detta är en talsekvens som påminner på Fibonacci talen, men istället för att börja med 0 och 1 och sen addera de två tidigare talen i sekvensen så börjar man med 0, 0 och 1, och sen adderar man de *tre* tidigare talen för att få nästa i sekvensen. De första 10 talen i sekvens är alltså:

$$0, 0, 1, 1, 2, 4, 7, 13, 24, 44, \ldots$$

Implementera ett predikat `tribonacci(N,T)` där `N` är ett tal och `T` det `N`:e Tribonacci talet. (3p)

Exempelanvändning:

```
?- tribonacci(0,T).
T = 0
?- tribonacci(1,T).
T = 0
?- tribonacci(2,T).
T = 1
?- tribonacci(9,T).
T = 44
```

(b) Haskellfunktionen `zip :: [a] -> [b] -> [(a, b)]` fungerar på följande sätt:

```
> zip [1,2,3] [4,5,6]
[(1,4),(2,5),(3,6)]
> zip [1,2,3] [4,5,6,7,8,9]
[(1,4),(2,5),(3,6)]
> zip [1,2,3] []
[]
> zip [] [1,2,3]
[]
```

Implementera motsvarande funktionalitet i Prolog som predikatet `zip(XS,YS,ZS)` där `XS` och `YS` är indatalistorna och `ZS` den unika utdatalistan. (3p)

Obs: notera vad som händer i Haskell om listorna har olika längd, Prologlösningen ska fungera på samma sätt. För full poäng ska även Prolog direkt avsluta sökningen när den hittat utdatalistan och inte försöka söka vidare efter lösningar.