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

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2023-08-14

- **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. Vad blir resultatet av om man kör följande Prologkod: `1 + 1 = 2.`

A. `true.`
B. `false.`
C. `X = 1 + 1, Y = 2.`
D. `1 + 1 = 2.`
E. `2 = 2.`

2. Vilket/Vilka påståenden nedan är sanna?

A. Tolkade (eng. *interpreted*) språk är alltid snabbare än kompilerade.
B. Java är ett statiskt typat språk.
C. Typklasser i Haskell är samma sak som klasser i Java.
D. Typfel hittas vid kompilering för statiskt typade språk.
E. En polymorf funktion är en funktion som tar in en funktion som argument.

3. Betrakta Haskellkodsnutten till höger, vad blir resultatet av `f [9,2,1,3]`?

A. 3
B. 4
C. 6
D. 10
E. 15

```
f :: [Int] -> Int
f [] = 0
f (x : xs) = length xs + f xs
```

4. Vad skrivs ut om man kör C-koden till höger?

A. `%d`
B. 20
C. 30
D. 600
E. Inget då koden leder till kompileringsfel

```
void fun(int *x) {
    *x = 30;
}

int main() {
  int y = 20;
  fun(&y);
  printf("%d", y);
}
```

5. Vilka av följande påståenden stämmer för JavaScript?

A. JavaScript är statiskt typat.
B. Variabler i JavaScript kan byta typ efter att man deklararerat dem.
C. JavaScript är en variant av Java.
D. Typfel hittas vid kompilering av JavaScript kod.
E. Kod skriven av JavaScript kan köras i webbläsare.

6. Vad gäller för Java koden till höger?

A. `Animal` är en superklass till `Cat`
B. `Animal` är en subklass till `Cat`
C. `Animal`'s konstruktor tar en sträng som argument
D. `Animal`'s konstruktor tar en sträng och ett heltal som argument
E. `age` och `legs` är instansattribut i `Cat`

```
class Cat extends Animal {
    public int age;
    public int legs;

    public Cat(String name,int age) {
        super(name);
        this.age = age;
        this.legs = 4;
    }
}
```

7. Vad är typen på Haskellfunktionen:

```
foo f xs = filter f xs ++ map f xs
```

A. `(a -> b) -> [a] -> [b]`
B. `(a -> Bool) -> [a] -> [Bool]`
C. `(a -> Bool) -> [a] -> [b]`
D. `(Bool -> Bool) -> [Bool] -> [Bool]`
E. `(a -> Int) -> [Bool] -> [Int]`

8. Hur många lösningar kommer Prolog generera för ett anrop till `p(X,Y)` givet koden nedan till höger?

A. 1
B. 2
C. 3
D. 4
E. 5

```
p(X,Y):- q(X), !, r(Y).

q(1).
q(2).

r(3).
r(4).
r(5).
```

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 calculate(int* p, int* q, int n)
```

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

Exempelanvändning:

Om man kör följande kodsnutt:

```
int main() {

  int x[5] = {1,2,3,4,5};
  int y[5] = {3,2,1,6,2};

  printf("%d\n", calculate(&x[0],&y[0],5));
  printf("%d\n", calculate(&x[0],&y[0],2));
}
```

så ska resultaten som skrivs ut vara `44` och `7` (dvs `1*3 + 2*2 + 3*1 + 4*6 + 5*2` och `1*3 + 2*2`).

(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 skapa klasser för att representera olika typer av filer på en dator.

(a) Skriv klassen `File` som ska ha *publika* instansattribut `name` för filnamnet och `ending` för filändelsen, båda av typ `String`. Klassen ska även ha en konstruktormetod som låter användaren sätta dessa samt en `toString` metod som skriver ut hela filnamnet med en punkt mellan namnet och filändelsen. (2p)

(b) Skriv en klass `Document` som ärver från `File`. Klassen ska ha ett *privat* instansattribut `content` av typ `String` samt en konstruktor som låter användaren sätta filnamnet och `content`. Filnamnet ska sättas genom att anropa konstruktorn i superklassen och filändelsen ska då sättas till `"doc"`. (2p)

(c) Skriv en klass `Song` som även den ärver från `File`. Klassen ska inte ha några instansattribut, men konstruktorn ska låta användaren ge bandnamn och låtnamn och sen anropa konstruktorn för superklassen med bandnamn och låtnamn sammanslagna med `" - "` mellan (se exemplet nedan). Filändelsen ska då även sättas till `"mp3"`. (1p)

Exempelkörning: om man testar klassen med koden

```
class uppgift10 {
    public static void main(String[] args) {
        File f = new File("exam","txt");
        System.out.println(f);

        Document d = new Document("solutions","TODO");
        System.out.println(d);

        Song s = new Song("Bruce Springsteen","The River");
        System.out.println(s);
    }
}
```

så ska utskriften bli:

```
exam.txt
solutions.doc
Bruce Springsteen - The River.mp3
```

11. Funktionell programmering i Haskell

(a) Skriv en funktion `calculate :: [Int] -> [Int] -> Int -> Int` som fungerar som C funktionen i uppgift 9 ovan. Så `calculate xs ys n` ska beräkna summan av produkterna av de `n` första talen i `xs` och `ys`. För full poäng får man inte använda några inbyggda Haskellfunktioner (så endast aritmetiska funktioner, rekursion, guards, och mönstermatchning är tillåtet). (4p)

Obs: om någon av listorna är kortare än `n` så ska funktionen inte krascha utan fungera som i sista exemplet nedan (dvs funktionen multiplicerar bara så länge det finns tal kvar i listorna).

Exempelanvändning:

```
> calculate [1,2,3,4,5] [3,2,1,6,2] 5
44
> calculate [1,2,3,4,5] [3,2,1,6,2] 2
7
> calculate [1,2,3,4,5] [3,2] 5
7
```

(b) Skriv `calculate` med hjälp av de inbyggda funktionerna `sum`, `take` och `zipWith`. (2p)
Dessa funktioner har följande typer

```
sum :: [Int] -> Int
take :: Int -> [a] -> a
zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
```

och fungerar på följande sätt:

```
> sum [1,2,3,4,5]
15
> take 3 [1,2,3,4,5]
[1,2,3]
> zipWith (+) [1,2,3,4,5] [3,2,1,6,2]
[4,4,4,10,7]
```

12. Logikprogrammering i Prolog

(a) Betrakta Haskellfunktionen nedan:

```
f :: [Int] -> [Int]
f [] = []
f (x:xs) = x : x : f xs
```

Implementera ett predikat `f(In,Out)` som relaterar en lista `In` med listan `Out` som innehåller samma element som resultatet av att köra Haskellfunktionen ovan med listan `In`. (2p)

(b) Skriv ett predikat `calculate(XS,YS,N,R)` där `R` är resultatet av att köra `calculate` funktionen ovan. För full poäng ska predikatet fungera som Haskell-funktionen i uppgift 11. (4p)
Exempelanvändning:

```
?- calculate([1,2,3,4,5],[3,2,1,6,2],5,C).
C = 44 .
?- calculate([1,2,3,4,5],[3,2,1,6,2],2,C).
C = 7 .
?- calculate([1,2,3,4,5],[3,2],10,C).
C = 7 .
```