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

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2026-04-15

- **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 skrivs ut av koden till höger?

A. `-2`
B. `-1`
C. `0`
D. `1`
E. `2`

```
int *p = malloc(5 * sizeof(int));
int *q = malloc(5 * sizeof(int));

for (int i = 0; i < 5; i++) {
  *(q + i) = i;
  *(p + i) = *q - i;
}

printf("%d", *(p + 2));
```

2. Hur många lösningar (d.v.s. tilldelningar till `X` och `Y`) kommer Prolog generera för ett anrop till `p(X,Y)` givet koden nedan?

A. 0
B. 1
C. 2
D. 3
E. 4

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

q(1).
q(2).

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

3. Vad blir resultatet av `f [(1,2),(3,4),(5,6)]` givet Haskellkoden nedan?

A. 0
B. 1
C. 2
D. 3
E. 4

```
f [] = 0
f ((x,_):xs) = x + g xs

g [] = 0
g ((_,y):xs) = y - f xs
```

4. Vilka av följande påståenden är sanna?

A. Variabler kan byta typ efter att de deklarerats i dynamiskt typade språk
B. Dynamiskt typade språk måste kompileras för att kunna köras
C. Dynamiskt typade språk är alltid snabbare än statiskt typade

D. JavaScript är dynamiskt typat

E. Java är dynamiskt typat

5. Givet Java-koden till höger, på vilket eller vilka sätt kan man skapa bilen `c` som är en Volvo 740 ägd av Elon Musk?

```
class Volvo extends Car {
   public String manufacturer;
   public String model;

   public Volvo(String model,String owner) {
     super(owner);
     this.manufacturer = "Volvo";
     this.model = model;
   }
}
```

A. `c = new Volvo("740")`

B. `c = new Volvo("740","Elon Musk");`

C. `c = new Volvo("Volvo","740","Elon Musk");`

D. `c = new Car("740","Elon Musk");`

E. `c = new Car("Volvo","740","Elon Musk");`

6. Vad blir resultatet av `f [1,2,3,4]` givet Haskellkoden nedan?

```
f [] = []
f (x:xs) = x : f (map (\y -> x + y) xs)
```

A. `[1,2,3,4]`

B. `[1,3,5,9]`

C. `[1,3,5,10]`

D. `[1,3,6,10]`

E. `[1,3,7,15]`

7. Givet C-koden till höger, vad är värdet på `x` när koden körts?

```
int x = 1;
int i = 1;
start:
    if (x > i)
        goto end;
    else
        x += x;
    i++;
    if (i <= 5)
        goto start;
end:
```

A. `0`

B. `1`

C. `2`

D. `3`

E. `4`

8. Vad är typen på Haskellfunktionen:

```
f x y = fst x : snd x : fst y : snd y
```

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

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

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

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

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

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**

Ett tal är *perfekt* om det är samma som summan av dess delare. Om vi tar 6 som exempel så är delarna 1, 2, och 3. Summerar vi dessa får vi $1 + 2 + 3 = 6$. Tar vi 12 istället så är delarna 1, 2, 3, 4 och 6, vilket ger $1 + 2 + 3 + 4 + 6 = 16 \neq 12$. Vi har alltså att 6 är perfekt, men 12 är inte det.

(a) Börja med att skriva en funktion med typ `int* divisors(int n)` som returnerar en pekare till början av en minnessekvens med delarna till `n`. Första värdet i sekvensen ska vara antal delare och sen ska

delarna följa. För full poäng får man inte ha nollor eller andra standardvärden i slutet av sekvensen utan det ska vara antal delare följt av exakt så många delare. Givet `6` ska alltså `divisors` returnera en pekare till en minnessekvens med `3,1,2,3` och givet `12` ska resultatet bli `5,1,2,3,4,6`. Kom ihåg att man inte kan dela med `0`. (3p)

(b) Skriv ett program som låter användaren skriva in tal och sen använder `divisors` för att avgöra om talet är perfekt eller inte. Programmet ska alltså göra följande:

i. Be användaren om ett tal m.h.a. `printf` och läsa in svaret med `scanf`.
ii. Anropa `divisors` med talet och spara resultatet på lämpligt sätt.
iii. Använd resultatet från `divisors` för att se om talet är perfekt eller inte.
iv. Skriv ut resultatet av testet till användaren (t.ex. genom att skriva ut ”The number is perfect!” och ”The number is not perfect!”).

För poäng måste `divisors` användas på lämpligt sätt. (2p)

10. Objektorienterad programmering i Java

Här ska ni skriva klasser för att representera (enkelt länkade) listor som innehåller `Integers`. Man ska kunna ta längden av listor med hjälp av en metod `length` och en lista är antingen `Cons` av ett heltal och en till lista, eller `Empty`, dvs tomma listan.

(a) Börja med att skriva en *abstrakt* klass `List` med en abstrakt metod `length`. (2p)

(b) Lägg till en klass `Empty` som ärver `List` och har en konstruktor som inte tar några argument. Implementera även `length` metoden för `Empty` listor. (1p)

(c) Lägg till en klass `Cons` som ärver `List` och vilken har en konstruktor som tar en `Integer` och en lista och sätter lämpliga instansattribut. Implementera även `length` metoden för `Cons` listor. (2p)

Exempelkörning: om man testar klasserna med koden

```
class uppgift10 {

  public static void main(String[] args) {
    // Tomma listan []
    List e = new Empty();

    // Listan [1,4,5]
    List l = new Cons(1,new Cons(4,new Cons(5,new Empty())));

    System.out.println(e.length());
    System.out.println(l.length());
  }
}
```

så ska utskriften bli:

```
0
3
```

11. Funktionell programmering i Haskell

(a) Skriv en funktion `sumFst :: [(Int,a)] -> Int` med hjälp av de inbyggda funktionerna

```
map :: (a -> b) -> [a] -> [b]
fst :: (a,b) -> a
sum :: [Int] -> Int
```

och vilken beräknar summan av de första elementen i listan. (1p)

Exempelanvändning:

```
> sumFst [(1,"hej"),(2,"hopp"),(3,"hej")]
6
```

(b) Skriv en funktion `prefix :: String -> String -> Bool` som returnerar `True` om första strängen är ett prefix till den andra strängen. (2p)

Exempelanvändning:

```
> prefix "hej" "hej"
True
> prefix "hej" "hejhopp"
True
> prefix "hej" "nej"
False
```

(c) Skriv en datatyp `Letter` som antingen är `Vowel` eller `Consonant`. Skriv även en funktion `typeOfLetter` av typ `Char -> Letter` som säger om bokstaven är en vokal (en av ”aeiouy”) eller konsonant (någon annan bokstav). Man kan anta att man bara får in bokstäver på engelska (d.v.s. inte åäö). (3p)

Tips: den inbyggda funktionen `elem :: Eq a => a -> [a] -> Bool` kan vara användbar.

Exempelkörning:

```
> typeOfLetter 'a'
Vowel
> typeOfLetter 'k'
Consonant
```

12. Logikprogrammering i Prolog

(a) Skriv `split(XS,N,YS,ZS)` som delar upp listan `XS` i två delar `YS` och `ZS` där `YS` har längd `N`. (2p)

Exempelanvändning:

```
?- split([1,2,3,4,5],2,YS,ZS).
YS = [1, 2],
ZS = [3, 4, 5].
```

(b) Skriv `evenOdd(L,E,O)` som funkar så att om `L` är indatalistan så är `E` alla värden på jämna index och `O` alla värden på udda index. Index räknas från noll. (2p)

Exempelanvändning:

```
?- evenOdd([3,2,1,5,7],E,O).
E = [3, 1, 7]
O = [2, 5].
```

(c) Vi kan använda konstanterna `cons(X,XS)` och `empty` i Prolog för att representera listor precis som vi gjorde i Java ovan. Skriv en relation `length_list(L,N)` så att `N` är längden på `cons`/`empty`-listan `L`. (2p)

Exempelanvändning:

```
?- length_list(empty,N).
N = 0.
?- length_list(cons(1,cons(4,cons(5,empty))),N).
N = 3.
```