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

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2025-04-23

- **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 är en abstrakt klass?

A. En klass utan några attribut.
B. En klass utan några metoder.
C. En klass som inte kan instansieras.
D. En klass som representerar en rekursiv datatyp.
E. Det finns inget som heter ”abstrakt klass” inom objektorientering.

2. Vilka påståenden nedan är sanna?

A. Tolkade (eng. *interpreted*) språk är alltid snabbare än kompilerade.
B. Haskell är ett statiskt typat språk.
C. JavaScript är en variant av Java som körs i webbläsaren.
D. Typklasser i Haskell är till för att göra det möjligt att skriva objektorienterad kod trots att Haskell är ett funktionellt språk.
E. Variabler kan byta typ efter de deklarerats i dynamiskt typade språk.

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

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

```
int x = 0;

for (int i = 5; i >= 0; i--)
    if (x >= i)
        goto end;
    else
        x += x;

end:
```

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

A. `%p`
B. `4` då vi endast skriver ut första siffran av `x`.
C. `x`
D. `42`
E. Det går inte att säga på förhand då `p` innehåller en godtycklig minnesadress.

```
int x = 42;
int *p;
p = &x;

printf("%p", p);
```

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 = Volvo("740","Elon Musk");`

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

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

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

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

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

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

q(1).
q(2).

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

A. 2

B. 3

C. 4

D. 5

E. 6

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

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

A. `[1,2,3]`

B. `[1,3,5]`

C. `[1,3,6]`

D. `[1,2,7]`

E. `[1,2,3,4,5]`

8. Vad är typen på Haskellfunktionen:

```
f x y = fst 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,b) -> (b,[c]) -> [c]`

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

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**

(a) Skriv en funktion `void swap(int* p,int *q)` som byter plats på talen som `p` och `q` pekar på. Så om `p` pekar på `42` och `q` på `58` kommer `swap(p,q)` leda till att `p` pekar på `58` och `q` på `42`. (1p)

(b) Skriv exempelkod som visar hur man gör så att `p` pekar på `42` och `q` på `58`, och sen anropas `swap` så att det de pekar på byter plats. (2p)

(c) Skriv en funktion `void swap_many(int* p, int* q, int n)` som tar in två pekare som pekar på två minnessekvenser med `n` tal och sen byter plats på alla värden i sekvenserna med hjälp av `swap`. Man kan använda sig av `swap` även om man inte löst den deluppgiften. (2p)

Exempelkörning:
Om man kör följande kodsnutt:

```
int list1[5] = {1,2,3,4,5};
int list2[5] = {-1,-2,-3,-4,-5};
swap_many(list1,list2,5);

for (int i = 0; i < 5; i++)
    printf("%d ",*(list1+i));
```

så ska resultatet vara

```
-1 -2 -3 -4 -5
```

10. Objektorienterad programmering i Java

På den här uppgiften ska man implementera länkade listor med en max-metod med hjälp av objektorientering. En länkad lista är en datastruktur som representerar listor där element antingen är slutet på listan eller ett värde och resten av listan.

(a) Skapa en abstrakt klass `LinkedList` med en abstrakt metod `max` som returnerar en `Integer`. Klassen `LinkedList` är tänkt att vara huvudklass för resten av klasserna i uppgiften. (2p)

(b) Skriv en klass `End` för att representera sista elementet i listan. Klassen ska ärva `LinkedList` och ha en konstruktor som låter en sätta ett instansattribut `value` av typ `Integer`. Klassen ska även implementera `max` metoden från superklassen på lämpligt sätt. (1p)

(c) Skriv en klass `Node` som representerar element som inte är slutet av listan. Den ska ha instansattribut `value` av typ `Integer` och `next` av typ `LinkedList`. Konstruktorna ska låta en sätta dessa. Den ska även implementera `max` metoden från huvudklassen på lämpligt sätt. (2p)

Exempelkörning: om man testar klassen med koden

```
public static void main(String[] args) {

    // Skapa listan med 1, 2 och 3
    LinkedList ex = new Node(1,new Node(2,new End(3)));

    System.out.println(ex.max());
}
```

så ska utskriften bli:

```
3
```

11. Funktionell programmering i Haskell

(a) Skriv en datatyp `Nucleotide` för att representera de fyra nukleotiderna `A`, `C`, `G`, `T`. (1p)

(b) Skriv funktionen `complement :: Nucleotide -> Nucleotide` som byter `A` mot `T` och `C` mot `G`, och viceversa. (1p)

(c) Vi kan nu skriva `type DNA = [Nucleotide]` så att DNA-sekvenser representeras som listor av nukleotider. Skriv en funktion `reverseComplement :: DNA -> DNA` som vänder sekvensen baklänges och applicerar `complement` på alla värden i sekvensen. (2p)

Exempelanvändning:

```
> reverseComplement [A,C,T,G,G,G,T,C,A,C]
[G,T,G,A,C,C,C,A,G,T]
```

(d) En k-mer är en delsträng av längd k i en DNA-sekvens. Skriv `kmers :: Int -> DNA -> [DNA]` som returnerar alla k-mers i en DNA-sekvens. För enkelhets skull kan man anta att k är mindre än längden på DNA-sekvensen. (2p)

Exempelanvändning:

```
> kmers 3 [G,T,A,G,A,G,C,T,G,T]
[[G,T,A],[T,A,G],[A,G,A],[G,A,G],[A,G,C],[G,C,T],[C,T,G],[T,G,T]]
> kmers 8 [G,T,A,G,A,G,C,T,G,T]
[[G,T,A,G,A,G,C,T],[T,A,G,A,G,C,T,G],[A,G,A,G,C,T,G,T]]
> kmers 10 [G,T,A,G,A,G,C,T,G,T]
[[G,T,A,G,A,G,C,T,G,T]]
```

12. Logikprogrammering i Prolog

(a) Skriv ett predikat `split(XS,N,Start,Middle,End)` som delar listan `XS` så att `Start` är de `N` första elementen, `End` är de `N` sista, och `Middle` är mitten av listan utan `Start` och `End`. För full pott måste man använda sig av snitt (`!`) på lämpligt sätt så att predikatet avslutas direkt när det hittat en lösning. För enkelhets skull får man anta att längden på `XS` är mer än `2 * N`. (3p)

Exempelanvändning:

```
?- split([1,2,3,4,5,6,7],2,Start,Middle,End).
Start = [1, 2],
Middle = [3, 4, 5],
End = [6, 7].
```

(b) Skriv ett predikat `permute_middle(XS,N,Out)` så att `Out` är listan `XS` med värdena i mitten omkastade, men de `N` första och sista värdena oförändrade. (3p)

Tips: Man får använda `split` även om man inte löst den uppgiften. Glöm inte bort det inbyggda predikatet `permutation(XS,YS)` som testar om `YS` är en permutation av `XS`, det kan var användbart.

Exempelanvändning:

```
?- permute_middle([1,2,3,4,5,6,7],2,Out).
Out = [1, 2, 3, 4, 5, 6, 7] ;
Out = [1, 2, 3, 5, 4, 6, 7] ;
Out = [1, 2, 4, 3, 5, 6, 7] ;
Out = [1, 2, 4, 5, 3, 6, 7] ;
Out = [1, 2, 5, 3, 4, 6, 7] ;
Out = [1, 2, 5, 4, 3, 6, 7] ;
false.
```