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

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2025-03-10

- **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. Givet Javascript/HTML kodsnutten `<button onclick="displayDate()">The time is?</button>`, vilka av påståendena nedan är sanna?

A. Funktionen `displayDate` kör alltid när webbsidan laddas.
B. För att köra `displayDate` kan man klicka på vilken knapp som helst på sidan.
C. För att köra `displayDate` kan man klicka på knappen där det står `The time is?`.
D. Koden kan inte köras då JavaScript-kod inte kan blandas med HTML.
E. Koden kan inte köras då det inte finns något event som heter `onclick`.

2. Vilket/vilka av följande påståenden beskriver en ”högre ordningens funktion”?

A. En funktion som är väldigt välskriven.
B. En funktion som tar in funktioner som argument.
C. En funktion som returnerar en funktion.
D. En funktion som kan ta argument av olika typer.
E. En funktion som läser in data från en fil.

3. Vad skrivs ut av koden till höger?

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

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

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

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

4. Vad gäller för Java-koden till höger?

A. Konstruktorn i `Car` tar en sträng som argument
B. `model` är en klassvariabel
C. `model` är en instansvariabel
D. `manufacturer` är en klassvariabel
E. `manufacturer` är en instansvariabel

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

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

5. Vad är typen på Haskellfunktionen `\f g x y -> f (g x) y` ?

A. `(a -> b) -> (b -> c) -> a -> b -> c`
B. `(a -> b -> c) -> (b -> c) -> a -> b -> c`
C. `(a -> b -> c) -> a -> b -> c`
D. `(a -> b -> c) -> (a -> b) -> a -> b -> c`
E. `(a -> b -> c) -> (d -> a) -> d -> b -> c`

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?

A. 2
B. 3
C. 4
D. 5
E. 6

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

q(1).
q(2).

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

7. 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) = f xs - y
```

8. Vilka av följande motsvarar en fas i en typisk implementation av ett statiskt typat programmeringsspråk?

A. Lexer
B. Typechecker
C. Organizer
D. Printer
E. Parser

Del 2: kodfrågor (22p totalt)

Var snäll och börja på ett nytt papper för varje uppgift i del 2. Lösningarna på deluppgifterna för varje programmeringsspråk kan skrivas på samma papper.

9. **Imperativ programmering i C**

Ett tal är *ymnigt* (eng. *abundant*) om summan av dess positiva delare (exklusive talet själv) är strikt större än talet själv. Om vi tar 12 som exempel så är delarna 1, 2, 3, 4, 6. Summerar vi dessa får vi 1+2+3+4+6 = 16, vilket är större än 12. Tar vi 8 istället så är delarna 1, 2 och 4, vilket ger 1 + 2 + 4 = 7 som är mindre än 8.

Vi har alltså att 12 är ymnigt, men 8 är inte det. På den här uppgiften ska ni skriva funktioner för att testa om tal är ymniga eller inte.

(a) Börja med att skriva en funktion med typ `int* divisors(int n)` som returnerar en pekare till början av en minnessekvens med alla tal `1 <= d < n` som delar `n`. För enkelhets skull ska ni allokera `n - 1` tal i minnet (t.ex. m.h.a. `malloc`) och om d inte delar n så sätter ni in en nolla. Givet `8` ska alltså `divisors` returnera en pekare till en minnessekvens med de sju talen `1,2,0,4,0,0,0`. För poäng måste man använda sig av `for`-loopar på den här uppgiften. (2p)

(b) Skriv om `divisors` till en funktion `int* divisors_goto(int n)` som bara använder sig av `goto` istället för `for`-loopar. (2p)

(c) Skriv en funktion `int is_abundant(int n)` som använder sig av `divisors` för att testa om `n` är ymnigt eller inte. Är talet ymnigt ska `1` returneras och `0` annars. (1p)

Exempelkörning:

Om man kör följande kodsnutt:

```
int* p1 = divisors(8);
for (int i = 0; i < 7; i++)
  printf("%d ",*(p1 + i));

printf("\n");

int* p2 = divisors_goto(8);
for (int i = 0; i < 7; i++)
  printf("%d ",*(p2 + i));

printf("\n%d\n%d\n",is_abundant(8),is_abundant(12));
```

så ska resultatet vara

```
1 2 0 4 0 0 0
1 2 0 4 0 0 0
0
1
```

10. Objektorienterad programmering i Java

Du har nyligen fått jobb på ett svenskt IT-bolag inom musikbranschen och din första uppgift på ditt nya jobb är att skriva klasser för att representera delar av deras musikkatalog.

(a) Skriv en klass `Length` som representerar låtlängder. Klassen ska ha publika instansattribut `min` och `sec` som representeras med hjälp av `Integers` och vilka sätts med en konstruktor. (1p)

(b) Lägg till en metod `add(Integer m,Integer s)` som lägger till `m` minuter och `s` sekund till längden som metoden anropas från. Kom ihåg att en minut har 60 sekunder. För att få resten vid heltalsdivison i Java använder man `%` (modulo) och för heltalsdivision används `/`, så `7 % 3 = 1` och `7 / 3 = 2`. (2p)

(c) Skriv en klass `Song` som representerar en låt. Klassen ska ha en konstruktor som låter användaren sätta de publika instansattributen `artist`, `name`, och `length`. Artistens namn (`artist`) och låtens namn (`name`), representeras som strängar. Låtens längd (`length`) representeras med hjälp av `Length` klassen ovan. (1p)

Klassen ska även ha en `toString` metod som returnerar en sträng på formen `artist - name (m,s)` (se nedan för exempel på hur det ska se ut). (1p)

Exempelkörning: om man testar klassen med koden

```
class uppgift10 {

    public static void main(String args[]) {
        Length l = new Length(3,33);
        Song the_ties_that_bind =
```

```
            new Song("Bruce Springsteen","The Ties That Bind",l);
        System.out.println(the_ties_that_bind);

        l.add(0,86); // The River is 86s longer than The Ties That Bind
        Song the_river = new Song("Bruce Springsteen","The River",l);
        System.out.println(the_river);

        l.add(-1,-40); // Subtracting 1m40s gives the length of Hungry Heart
        Song hungry_heart = new Song("Bruce Springsteen","Hungry Heart",l);
        System.out.println(hungry_heart);
    }
}
```

så ska utskriften bli:

```
Bruce Springsteen - The Ties That Bind (3,33)
Bruce Springsteen - The River (4,59)
Bruce Springsteen - Hungry Heart (3,19)
```

11. Funktionell programmering i Haskell

(a) Skriv en funktion `conjunctify :: [String] -> String` som givet en lista av strängar returnerar en sträng där komma satts in mellan alla ord utom de två sista där `and` satts in istället. (3p)

Exempelkörning:

```
> conjunctify ["Flour","milk","butter","eggs","salt"]
Flour, milk, butter, eggs and salt
> conjunctify ["Flour","milk"]
Flour and milk
> conjunctify ["Flour"]
Flour
```

(b) Skriv funktionen `iterate :: (a -> a) -> a -> Int -> [a]` som returnerar en lista av längd `n` där elementet på index `i` är `f` applicerad `i` gånger på `x`. Listan ska alltså ha formen `[x,f(x),f(f(x)),...]` där det sista elementet är `f` applicerad `n-1` gånger på `x`. (3p)

Obs: man får inte använda den inbyggda funktionen `iterate`.

Exempelkörning:

```
> iterate (\x -> 2 * x) 1 10
[1,2,4,8,16,32,64,128,256,512]
```

12. Logikprogrammering i Prolog

(a) Skriv ett predikat `suffix(Suff,S)` som returnerar `true` om strängen `Suff` är ett suffix till strängen `S` och `false` annars. För full poäng ska sökningen avslutas direkt efter att man fått `true`, så lösningen bör använda snitt (d.v.s. `!`) på lämpligt sätt. (3p)

Exempelkörning:

```
?- suffix("hej","hej").
true.
?- suffix("hej","hejhopp").
false.
?- suffix("hopp","hejhopp").
true.
```

Tips: konvertera först strängarna till listor med hjälp av `string_to_list(S,L)` (här är `S` en sträng och `L` en lista).

(b) Ett binärt träd med tal kan representeras i Prolog som antingen ett löv `leaf(X)` där `X` är ett tal eller en förgrening `branch(X,L,R)` där `X` är ett tal och `L` och `R` är underträd. Två exempel på binära träd på detta format är:

```
branch(2, leaf(4), leaf(8)).
branch(2, branch(3, leaf(10), leaf(8)), leaf(4))
```

Skriv ett predikat `max_tree(T,M)` där `T` är ett binärt träd och `M` är det maximala värdet i trädet. (3p)
Tips: det kan underlätta att skriva ett hjälppredikat `max3(X,Y,Z,M)` så att `M` är max av `X`, `Y` och `Z`.
Exempelkörning:

```
?- max_tree(branch(2,leaf(4),leaf(8)),M).
M = 8 .
?- max_tree(branch(2,branch(3,leaf(10),leaf(8)),leaf(4)),M).
M = 10 .
```