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

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2024-04-17

- **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 typen på Haskellfunktionen `foo f g xs = map (g . f) xs`?

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

2. Vilka av följande påståenden är sanna?

A. JavaScript stödjer händelsestyrd programmering.
B. Kod skriven i JavaScript kan bara köras om man först kompilerar den.
C. Man kan bara skriva loopar med hjälp av rekursion i JavaScript.
D. Variabler i JavaScript kan byta typ efter att de deklarerats.
E. JavaScript är dynamiskt typat.

3. Vad pekar `p` på när man kört koden till höger?

A. `1, 2, 3, 4, 5`
B. `5, 4, 3, 2, 1`
C. `1, 2, 3, 2, 1`
D. `5, 4, 3, 4, 5`
E. `-3, -2, -1, 0, 1`

```
int p[5] = {1,2,3,4,5};

for (int i = 0; i < 5; i++)
    *(p + i) = *(p + (4 - i));
```

4. Hur många möjliga lösningar hittas när man kör `f([1,2,3],YS)` givet Prologkoden till höger?

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

```
f(XS,YS) :-
    append(ZS,WS,XS),
    append(WS,ZS,YS).
```

5. Vad gäller för Javakoden till höger?

A. Konstruktorn i `Car` tar två strängar som argument

B. `manufacturer` är en klassvariabel

C. `manufacturer` är en instansvariabel

D. `model` är en klassvariabel

E. `model` är en instansvariabel

```
class Volvo extends Car {
   public static String manufacturer = "Volvo";
   public String model;

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

6. Vad blir resultatet av `foo 5 (+1) 0` givet koden till höger?

A. `[0,1,2,3,4]`

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

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

D. `[1,2,4,8,16]`

E. `[]`

```
foo :: Int -> (a -> a) -> a -> [a]
foo 0 _ _ = []
foo n f x = f x : foo (n - 1) f (f x)
```

7. Vilket/Vilka påståenden nedan är sanna?

A. Typklasser i Haskell är samma sak som klasser i Java.

B. Java är ett statiskt typat språk.

C. Variabler kan byta typ efter att de deklarerats i dynamiskt typade språk.

D. Typfel hittas vid kompilering för statiskt typade språk.

E. En polymorf funktion är en funktion som stödjer värden av olika typer.

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

A. 0

B. 1

C. 2

D. 3

E. 4

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

q(1).
q(2).

r(1).
r(2).
r(3).
```

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**

Fibonacci-sekvensen är en talsekvens som börjar med 0 och 1, och nästa tal i sekvensen fås sedan genom att addera de två tidigare talen i sekvensen. De första 10 talen i sekvens är alltså:

0 1 1 2 3 5 8 13 21 34 ...

På ordinarie tentan skulle man skriva olika funktioner i C för att beräkna sekvensen. Här ska ni skriva fler såna funktioner.

(a) Skriv funktionen `int fib_for(int n)` som returnerar det `n`:e Fibonacci-talet med hjälp av en `for`-loop. Lösningen måste vara skriven med en `for`-loop för poäng. (2p)

(b) Skriv funktionen `int fib_goto(int n)` som returnerar det `n`:e Fibonacci-talet med hjälp av `goto`. För poäng får man inte använda `for`, `while`, `do-while`, eller rekursion, utan det är `goto` som gäller. (2p)

(c) Istället för att beräkna ett specifikt tal i sevensen kan det vara mer effektivt att beräkna de `n` första talen i sekvensen. Nedan följer ett försök till en lösning inspirerad av facit till ordinarie tenta:

```
int* fib_numbers(int n) {
  int* f = malloc(n * sizeof(int));

  // Initialize two first values
  *f = 0;
  f++;
  *f = 1;
  f++;

  // Loop to get the rest
  for (int i = 0; i < n-2; i++) {
    *f = *(f - 1) + *(f - 2);
    f++;
  }

  return f;
}
```

Tanken är att funktionen ska returnera en pekare till början av de `n` första talen i sekvensen, men koden innehåller en bugg som gör att detta inte är fallet. Vad är buggen? (1p)

Exempelanvändning:

Om man kör följande kodsnutt:

```
int main() {
  for (int i = 0; i < 10; i++)
    printf ("%d ", fib_for(i));
  printf("\n");

  for (int i = 0; i < 10; i++)
    printf ("%d ", fib_goto(i));
  printf("\n");

  int* f = fib_numbers(10);
  for (int i = 0; i < 10; i++)
    printf ("%d ", *(f + i));
}
```

så ska resultatet vara

```
0 1 1 2 3 5 8 13 21 34
0 1 1 2 3 5 8 13 21 34
0 1 1 2 3 5 8 13 21 34
```

10. Objektorienterad programmering i Java

Den här uppgiften går ut på att representera sten-sax-påse spel i Java. Reglerna är som vanligt att papper slår sten, sten slår sax, sax slår papper, och vid samma blir det oavgjort. Klassen `RPS` ska innehålla följande saker: (5p)

- Två privata instansvariabler `player1` och `player2` av typ `String`.
- En privat metod `is_valid(String s)` som returnerar en `Boolean` som är sann om `s` är `"Rock"`, `"Paper"`, eller `"Scissors"`, och falsk annars.
- En konstruktor som tar in två strängar och sätter `player1` och `player2` om båda strängarna är giltiga, dvs om `is_valid` returnerar `true` för båda strängarna. Om indata är ogiltig kan man göra vad man vill; det viktiga är att rätt sak görs om indata är giltig.
- En publik metod `winner` som returnerar en sträng vilken är `"Player 1"` om `player1` slår `player2`, `"Player 2"` om `player2` slår `player1`, och `"Draw"` annars.

Exempelkörning: om man testar klassen med koden

```
RPS play1 = new RPS("Rock","Paper");
RPS play2 = new RPS("Scissors","Rock");
RPS play3 = new RPS("Scissors","Paper");
RPS play4 = new RPS("Paper","Paper");

System.out.println(play1.winner());
System.out.println(play2.winner());
System.out.println(play3.winner());
System.out.println(play4.winner());
```

så ska utskriften bli:

```
Player 2
Player 2
Player 1
Draw
```

11. Funktionell programmering i Haskell

(a) Skriv `applyOn :: Bool -> (a -> a) -> (a,a) -> (a,a)` som fungerar så att `applyOn b f p` applicerar funktionen `f` på första elementet i paret `p` om `b` är `True` och på det andra om `b` är `False`. (2p)

Exempelanvändning:

```
> applyOn True (*2) (2,3)
(4,3)
> applyOn False (*2) (2,3)
(2,6)
```

(b) Skriv `vowels :: String -> String` som returnerar alla vokaler i indatasträngen. Man behöver bara stödja små bokstäver och engelska vokaler, dvs "aeiouy". (2p)

Exempelanvändning:

```
> vowels "hello"
"eo"
```

(c) Skriv en funktion `maxPairs :: [(Int,Int)] -> [Int]` som returnerar en lista med de största värdena i varje par given en lista av par. (2p)

Exempelanvändning:

```
> maxPairs [(2,3),(5,5),(6,2)]
[3,5,6]
```

12. Logikprogrammering i Prolog

(a) Skriv ett predikat `max_pairs(XS,YS)` så att `YS` är maxvärdet för varje par i `XS`. (2p)

Exempelanvändning:

```
?- max_pairs([(2,3),(5,5),(6,2)],YS).
YS = [3, 5, 6].
```

(b) Skriv ett predikat `prefix(P,S)` som returnerar `true` om strängen `P` är ett prefix till strängen `S` och `false` annars. (2p)

Exempelanvändning:

```
?- prefix("hej","hej").
true.
?- prefix("hej","hejhopp").
true.
?- prefix("hej","nej").
false.
```

Tips: det kan underlätta att konvertera strängarna till listor med hjälp av `string_to_list`.

(c) Skriv ett predikat `sign(X,S)` som associerar ett tal `X` med `negative`, `zero`, eller `positive` beroende på om det är mindre än, lika med, eller större än noll. För full poäng måste man använda *snitt* så att bara en lösning returneras direkt i de olika fallen. (2p)

Exempelanvändning:

```
?- sign(-2,S).
S = negative.
?- sign(0,S).
S = zero.
?- sign(4,S).
S = positive.
```