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

Tentamensskrivning i
DA4003 Programmeringsparadigm
7.5 hp
2022-04-21

- **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 A på ett svarspapper. Varje fråga på del 1 är värd 1 poäng.

1. Vad returnerar en ”lexer”?

A. En exekverbar fil.
B. En lexikografiskt sorterad lista av ord.
C. Rad och kolumnnummer för alla ord i en fil.
D. En lista med tokens.
E. Ett syntaxträd.

2. Vad är typen på Haskellfunktionen:

```
f x y z = (x,y) : snd z
```

A. `(a,b) -> c -> [(a,b)]`
B. `(a,[b]) -> [b] -> (a,[b])`
C. `a -> b -> (c, d) -> ((a, b), c)`
D. `a -> b -> (c, [(a, b)]) -> [(a, b)]`
E. Ingen då koden leder till ett typfel.

3. Vilka påståenden nedan är sanna?

A. JavaScript är dynamiskt typat.
B. Ett språk kan inte vara både objektorienterat och funktionellt.
C. Ett tolkat (eng. *interpreted*) språk är alltid snabbare än ett kompilerat (eng. *compiled*).
D. Arrayer kan returneras från funktioner i C genom att man returnerar en pekare till början av arrayen.
E. Komma ”,” används för konjunktion (”och”) i Prolog

4. Vad skrivs ut av kodsnutten till höger?

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

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

int *p = &x[0];

printf("%d", x[*(p+3)]);
```

5. Vilket/vilka Prolog kodsnuttar fångar påståendet att ”If the sky is blue, then everyone likes it”?

A. `sky(blue) :- everyone(likes).`
B. `likes(X, sky) :- blue(sky).`
C. `blue(sky), likes(X).`
D. `blue(sky) :- likes(X,sky).`
E. Påståendet går inte att implementera i Prolog då Prolog inte har if-satser.

6. Vad gäller för Java koden till höger?

A. `Complex` är en klass med en konstruktor som tar två doubles.
B. `Complex` är en abstrakt klass.
C. `conjugate` är en metod.
D. `real` och `imag` är instansvariabler.
E. `r` och `i` är instansvariabler.

```
class Complex {
    double real, imag;

    public Complex(double r, double i) {
        real = r;
        imag = i;
    }

    public void conjugate() {
        imag *= -1;
    }
}
```

7. Vilka påståenden gäller för *högre ordningens funktioner*?

A. En högre ordningens funktion är en funktion som kan hantera godtyckligt stora tal.
B. En funktion som anropar en annan funktion är en högre ordningens funktion.
C. En funktion som tar in en funktion som parameter är en högre ordningens funktion.
D. En `for` loop är en typ av högre ordningens funktion.
E. En funktion som returnerar en funktion är en högre ordningens funktion.

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

A. 4
B. 6
C. 10
D. 15
E. Ingenting då vi fastnar i en oändlig loop.

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

Del 2: kodfrågor (12p totalt)

Var snäll använd ett papper till varje uppgift i del 2.

9. **Imperativ programmering i C** (5p)

Betrakta följande två C funktioner som beräknar längden på en sträng och vänder en sträng baklänges:

```
int length(char* s) {

  int c = 0;

  while (s[c] != '\0')
    c++;

  return c;

}
```

```
void reverse(char* s) {

  int l = length(s);

  for (int i = 0; i < l/2; i++) {
    char temp = s[i];
    s[i] = s[l - i - 1];
    s[l - i - 1] = temp;
  }
}
```

Uppgiften är nu att skriva om kodsnuttarna enligt instruktionerna nedan. Deluppgifterna (a)–(c) är oberoende av varandra, men bör skrivas på samma papper.

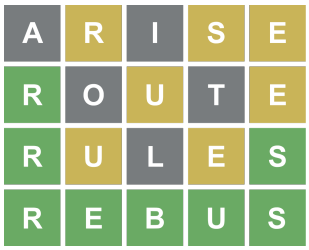
(a) Skriv om `length` så att den använder sig av en `for` loop istället för en `while` loop. (1.5p)

(b) Skriv om `length` så att den använder sig av `goto` istället för en `while` loop. (1.5p)

(c) Skriv om `reverse` så att den använder sig av pekararitmetik istället för array notation. (2p)

10. Objektorienterad programmering i Java

Har du dragits med i Wordle-yran? Wordle är ett online-spel där man på ett fåtal gissningar ska hitta det hemliga ordet, alltid 5 bokstäver långt. Varje gång man gissar får man återkoppling på vad som blivit rätt och fel: rätt bokstav på rätt plats blir markerat grönt och rätt bokstav på fel plats blir gulfärgad (se figur 1).



Figur 1: Wordle exempel från Wikipedia (”REBUS” är korrekt ord).

Den här uppgiften går nu ut på att implementera delar av en `Wordle` klass i Java.

(a) Implementera klassen så att den har en konstruktor som tar ett hemligt ord av typ `String` som parameter. Detta ord ska sparas i ett *privat* instansattribut. (2p)

(b) Lägg sedan till en metod `guess_word(String guess)` som returnerar en sträng med “färger”: `G` (för ”Green”) om en bokstav är korrekt gissad, `Y` (för ”Yellow”) när bokstaven finns i ordet men gissades på fel position, och `X` för fel bokstav. (3p)

För enkelhets skull ändrar vi spelet så att vi inte räknar antalet förekomster av en bokstav, så om det hemliga ordet är `KALLT` och gissningen är `BASTA` blir återkopplingen `XGXYY`, även om det bara är ett `A` som är korrekt. Den här förenklingen gör spelet mer irriterande, men det har vi överseende med här.

Man kan även anta att alla ord som användaren gissar är skrivna med versaler (stora bokstäver) och består av 5 bokstäver. Man behöver inte heller kolla så att användaren skriver in ett riktigt ord på svenska, utan metoden ska bara jämföra strängarna och returnera en sträng med `G`, `Y` och `X` enligt instruktionerna ovan.

Tips: följande `String` metoder kan vara användbara (men uppgiften går att lösa utan dem):

- `s.charAt(i)`: hämta ut bokstaven på index `i` i strängen `s` (räknat från `0`).
- `s.substring(i,j)`: plocka ut delsträngen mellan index `i` och `j` ur `s`.
- `s1.contains(s2)`: kolla om `s1` innehåller `s2` som delsträng.

Exempelanvändning: Givet följande testprogram:

```
class uppgift10 {
    public static void main(String[] args) {
        Wordle w = new Wordle("BASTU");
        System.out.println(w.guess_word("RIPOR"));
        System.out.println(w.guess_word("BANAN"));
        System.out.println(w.guess_word("BAMBU"));
        System.out.println(w.guess_word("BASTU"));
    }
}
```

så ska utdata bli:

```
XXXXX
GGXYX
GGXYG
GGGGG
```

11. Funktionell programmering i Haskell

Om man implementerar ett kortspel i Haskell behöver man ha en datatyp för att representera ett kort. I en vanlig kortlek har varje kort en färg (eng. *suit*). Detta kan implementeras som följande datatyp:

```
data Suit = Hearts | Diamonds | Clubs | Spades
  deriving (Show,Eq)
```

Ett spelkort har även en valör (eng. *rank*) vilket antingen är ett numeriskt värde mellan 2 och 10, ett ess (eng. *ace*), en knekt (eng. *jack*), en dam (eng. *queen*) eller en kung (eng. *king*). Uppgiften är nu att:

(a) Skriv en datatyp `Rank` för att representera ett spelkorts valör. Man behöver här inte ta hänsyn till att de numeriska korten måste vara mellan 2 och 10. (2p)

(b) Skriv en funktion `isValidRank :: Rank -> Bool` som kollar om en valör är giltig (dvs om det är ett numeriskt kort med värde mellan 2 och 10, ett ess, en knekt, en dam, eller en kung). (2p)

(c) Skriv en konstant `fullDeck :: [(Suit,Rank)]` vilken innehåller alla 52 spelkort i en vanlig kortlek. För full poäng får man inte hårdkoda hela kortleken genom att skriva alla 52 kort för hand. (2p)
Tips: skriv lämpliga hjälpfunktioner.

12. Logikprogrammering i Prolog

(a) På ordinarie tentan 2022-03-14 representerades binära träd med tal 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(4), leaf(8)), leaf(10)).
```

Skriv ett predikat `inc_tree(T1,T2)` där `T2` är ett binärt träd där alla värden `T1` har ökats med 1. (3p)
Exempelkörning:

```
?- inc_tree(branch(2, leaf(4), leaf(8)),T).
T = branch(3, leaf(5), leaf(9)).
?- inc_tree(branch(2, branch(3, leaf(4), leaf(8)), leaf(10)),T).
T = branch(3, branch(4, leaf(5), leaf(9)), leaf(11)).
```

(b) Skriv ett predikat `list_delete(X,L1,L2)` där `L2` är listan `L1` där alla `X` har tagits bort. (3p)
Exempelkörning:

```
?- list_delete(2,[1,2,3],XS).
XS = [1, 3] .
?- list_delete(2,[1,3],XS).
XS = [1, 3] .
?- list_delete(2,[2,2,2,2,2],XS).
XS = [] .
```