Bedömningsmall för tentamen EDAB05 Programmering, grundkurs

2026-01-07, 8:00-13:00

Hjälpmedel: Snabbreferenser för Scala och Java.

Checklista före rättning

Kolla att:

- antal inlämnade blad stämmer med omslaget,
- anonymkoden på varje inlämnat blad stämmer med omslaget,

annars kontakta rättningsansvarig som i sin tur kontaktar studierektor som i sin tur kontaktar Tentamensavdelningen, för att reda ut var sådana allvarliga fel har uppstått.

Generella bedömningsriktlinjer

Del A: Uppgift 1. Varje rad kan ge max 2p.

- Typerna och värdena ska skrivas som det står i lösningstabellen för full poäng. Dock får svaret skilja sig vad gäller i betydelselös användning av blanktecken och radbrytningar. Citationstecken runt strängar är ok men behövs inte. Om värdet är rätt, så när som på mindre detaljer i `toString`-representationen, till exempel `42.0` i stället för `42`, ges inget avdrag.
- Om rätt svar är ett kryss i kolumn 3 eller 4, men man kryssat för kompileringsfel när det ska vara exekveringsfel, eller tvärt om, eller om man kryssat i båda, ges -1 i avdrag.
- Om man svarat med fel typ men rätt värde ges -1 i avdrag.
- Ofullständig generisk typ, t.ex. `Option` istf. `Option[String]` ger -1 i avdrag.
- Om man svarat med rätt typ men fel värde ges -1 i avdrag.
- Om både värde och typ är fel ges -2 i avdrag.
- Om man svarat med typ/värde *och* kryss i kolumn 3 och/eller 4 på samma rad ges -2 i avdrag.
- Förväxling mellan `'` eller `"` eller glömda citattecken ger inget avdrag.
- Förväxling mellan stora och små begynnelsebokstäver t.ex. `True` istf. **`true`** ger inget avdrag.

Del B:

- Totala poängen för en uppgift kan inte bli negativ; om alla avdrag för en uppgift blir mer än maxpoängen ges 0 poäng.
- Enkla misstag som är lätta att åtgärda ger -1 i avdrag eller -2 i avdrag, beroende på hur allvarligt felet är.
- Allvarliga misstag eller stora ofullständigheter ger upp till *–maxpoäng* i avdrag, beroende på hur mycket av koden som måste skrivas om för att det ska fungera och vara begripligt.

Del A. Uppgift 1. Evaluera uttryck. Totalt max 20p.

Efter init. av:	Vid kompile-ringsfel sätt kryss.	Vid exekve-ringsfel sätt kryss.	Ange statisk **typ** som kompilatorn härleder om ej kompilerings- eller körtidsfel.	Ange det **värde** som tilldelas vid exe-kvering, med samma format som vid utskrift av värdets `toString`, om ej kompilerings- eller körtidsfel.
~~`u0`~~			~~`Double`~~	~~`42.0`~~
`u1`		**X**		
`u2`			`Boolean`	`true`
`u3`	**X**			
`u4`			`Int`	`2`
`u5`			`Int`	`1`
`u6`			`Int`	`3`
`u7`	**X**			
`u8`		**X**		
`u9`			`Option[Tomte]`	`None`
`u10`		**X**		

Rätta en rad i taget enligt nedan och sätt minuspoäng vid felaktigt svar i marginalen, t.ex. -1 eller -2. Räkna ut totala poängen och resultatet inringat längst ner på sidan direkt i skrivningshäftet. Varje rad kan ge max 2p.

- Typerna och värdena ska skrivas som det står i tabellen ovan för full poäng. Dock får svaret skilja sig vad gäller i betydelselös användning av blanktecken och radbrytningar. Om värdet är *nästan helt rätt*, så när som på obetydliga detaljer i `toString`-representationen, till exempel `10` i stället för `10.0` för ett värde av `Double`-typ, ges inget avdrag.
- Om rätt svar är ett kryss i någon av kolumnerna för kompilerings- eller exekveringsfel, men man kryssat för inkorrekt typ av fel, ges -1 i avdrag.
- Om man helgarderat med två kryss på samma rad ges -2 i avdrag.
- Om man svarat med fel typ men rätt värde ges -1 i avdrag.
- Om man svarat med rätt typ men fel värde ges -1 i avdrag.
- Om både värde och typ är fel ges -2 i avdrag.
- Om man svarat med typ/värde *och* kryss i någon av kolumnerna för kompilerings- eller exekveringsfel på samma rad ges -2 i avdrag.

Del B. Uppgift 2–4. Totalt max 80p.

Generella avdragsriktlinjer för del B:

- Totala poängen för en uppgift kan inte bli negativ; om alla avdrag för en uppgift blir mer än maxpoängen ges 0 poäng.
- Olika formateringsvarianter som skiljer sig från mönsterlösning som ej påverkar funktionen och ej allvarligt försvårar läsningen, t.ex. extra (klammer)parentespar eller radbrytningar, ger inga avdrag.
- Felaktigt matchade (klammer)parentespar, felaktig indentering, eller glömda nödvändiga (klammer)parenteser ger -1 i avdrag.
- Felaktigt avskriven metodsignatur från specifikationen ger inget avdrag om inte den felaktiga signaturen innehåller påhittade parametrar som används i lösningen; då ges -1 i avdrag per påhittad parameter och eventuella extra avdrag för att efterfrågad lösning saknas.
- Om nödvändigt semikolon saknas (t.ex. mellan flera satser på samma rad) ges -1 i avdrag.
- Användning av `try` eller Try som fångar alltför generella undantag ger, t.ex. `case e: Exception` i stället för `IndexOutOfBoundsException` ger $--1$ i avdrag
- Felaktig användning av tom parameterlista (om ej ska finnas eller om saknas) ger -1 i avdrag.
- Mindre fel i likhet med föregående två punkter där det framgår tydligt vad som egentligen avses ger -1 i avdrag.
- Ej användning av befintliga abstraktioner, egen onödig kod som redan finns i andra metoder: -1 i avdrag + ev. fel i onödig kod.
- Deklaration av `val` när det måste vara `var` för att koden ska fungera ger -2 i avdrag.
- Förlorat värde, t.ex. uttryck som räknar ut ngt viktigt men som aldrig sparas, ger -2 i avdrag.
- Deklaration av `var` när det skulle kunna vara en `val` och fungera lika bra ger -1 i avdrag.
- Onödigt krångliga booleska uttryck, t.ex. `quit = if (uttryck) true else false` i stället för bara `quit = uttryck`, ger 1p avdrag.
- Onödiga typannoteringar ger inget avdrag.
- Enstaka onödiga `this` ger inga avdrag
- Tillägg av element i samling på fel sätt, t.ex. `+:` i stället för `:+` eller liknande, ger -1 i avdrag.
- Användning av `return` ger -2 i avdrag.
- Användning av metoden `length` på samling som ej är en sekvens, t.ex `Set` el. `Map`, ger -1 i avdrag.
- Enkla misstag som är lätta att åtgärda ger -1 eller -2 i avdrag, beroende på hur allvarligt felet är.
- Allvarliga misstag eller stora ofullständigheter ger från -3 upp till *–maxpoäng* i avdrag, beroende på hur mycket av koden som måste skrivas om för att det ska fungera.
- Om precis samma typ av fel förekommer inom *samma* deluppgift (implementerad medlem) mer än en gång kan du välja att bara göra avdrag en gång vid första förekomsten, *om* detta ger en rimligare totalpoäng för deluppgiften. Om du bedömer att detta är rimligt, skriv då *– Samma fel* i kanten för att indikera att avdraget redan är gjort. I vissa speciella fall kan även avdrag för samma fel göras över flera implementerade medlemmar eller till och med över en hel uppgift eller för hela

skrivningen, under förutsättning att felet är trivialt, t.ex. för systematiska mindre syntaxfel så som onödiga semikolon vid radslut, konsekvent glömda `}` `)` **`then`** **`do`** eller upprepad användning av **`return`**.

- Dokumentationskommentarer, paketdeklarationer och importsatser i given kod behöver ej upprepas i lösningen. Om ytterligare importsatser behövs men helt saknas ska -1 i avdrag ges, men smärre felaktigheter i själva sökvägen ger inga avdrag. Om en fullständig sökväg anges direkt på plats (i stället för import), ges inget avdrag om sökvägen har smärre felaktigheter. Dock är det viktigt att distinktionen mellan `collection.mutable` och `collection.immutable` blir rätt och sådana felaktigheter ger -2 i avdrag.

Endast delarna markerade med `// Xp` ingår i bedömningen, där `X` är maximala poängen för respektive del.

Uppgift 2. `MutableMatrix`. (10p)

```
1  package solution
2
3  class MutableMatrix[T](val nRows: Int, val nCols: Int, val defaultValue: T):
4    require(nRows > 0 && nCols > 0, s"nRows and nCols must be non-zero")
5
6    import collection.mutable.ArrayBuffer
7
8    private val matrix: ArrayBuffer[ArrayBuffer[T]] =
9      ArrayBuffer.fill(nRows):
10       ArrayBuffer.fill(nCols)(defaultValue)
11
12   def apply(row: Int, col: Int): Option[T] = //tot 6p
13     if row >= 0 && row < nRows && col >= 0 && col < nCols
14     then Some(matrix(row)(col))
15     else None
16     // alternativ lösning 1:
17     //   matrix.lift(row).flatMap(_.lift(col))
18     //
19     // alternativ lösning 2:
20     //   if matrix.indices.contains(row) && matrix(row).indices.contains(col)
21     //   then Some(matrix(row)(col))
22     //   else None
23
24   def update(row: Int, col: Int, value: T): Unit = //tot 4p
25     if row >= 0 && row < nRows && col >= 0 && col < nCols
26     then matrix(row)(col) = value
27     else ()
28
29   def reset(row: Int, col: Int): Unit = update(row, col, defaultValue)
30
31 end MutableMatrix
```

-

Uppgift 3. `Terminal`. (33p)

```
1  package solution
2
3  class Terminal(val nRows: Int, val nCols: Int):
4    import Terminal.*
5
6    private val buffer = MutableMatrix(nRows, nCols, CharData.Blank)
7
8    private def printBlankAt(row: Int, col: Int): Unit =
9      moveCursor(row, col)
10     print(' ')
11
12   private def printCharDataAt(row: Int, col: Int): Unit =
13     moveCursor(row, col)
14     val cd = buffer(row, col).get
15     if !cd.isHighlight then print(cd.char)
16     else print(s"${InvertBackground}${cd.char}${Reset}")
17
18   def clear(): Unit = //tot 5p
19     for
20       row <- 0 until buffer.nRows
21       col <- 0 until nCols
22     do buffer.update(row, col, CharData.Blank)
23     print(s"$ClearScreen$MoveHome")
24
25   def getChar(row: Int, col: Int): Char = //tot 3p
26     buffer(row, col).getOrElse(CharData.Blank).char
27
28   def putChar(row: Int, col: Int, c: Char, isHighlight: Boolean): Unit = //tot 7p
29     if row >= 0 && row < nRows && col >= 0 && col < nCols then
30       buffer.update(row, col, CharData(c, isHighlight))
31       printCharDataAt(row, col)
32     else ()
33
34   def get(row: Int, fromCol: Int = 0, untilCol: Int = nCols): String = //tot 3p
35     val chars = for col <- fromCol until untilCol yield getChar(row, col)
36     chars.mkString
37
38   def put(text: String, row: Int, col: Int = 0, isHighlight: Boolean): Unit = //tot 4p
39     for i <- text.indices if col + i < nCols do
40       putChar(row, col + i, text(i), isHighlight)
41
42   def highlight(row: Int, col: Int): Unit = //tot 2p
43     putChar(row, col, getChar(row, col), true)
44
45   def scroll(): Unit = //tot 9p
46     for row <- 0 until nRows do
47       for col <- 0 until nCols do
48         if row < nRows - 1 then
49           buffer.update(row, col, buffer(row + 1, col).get)
50           printCharDataAt(row, col)
51         else
52           buffer.reset(row, col)
```

```
53            printBlankAt(row, col)
54        end for
55      end for
56
57  object Terminal:
58    case class CharData(char: Char, isHighlight: Boolean = false)
59
60    object CharData:
61      val Blank = CharData(' ')
62
63    val EscapeCode        = 27.toChar           // specialtecken som föregår styrkoder
64    val ClearScreen       = s"${EscapeCode}[2J" // terminalen rensas
65    val MoveHome          = s"${EscapeCode}[H"  // nästa utskrift sker överst till vänster
66    val InvertBackground  = s"${EscapeCode}[7m" // påbörjar markerad text
67    val Reset             = s"${EscapeCode}[0m" // avslutar markerad text
68
69    def width(default: Int = 60) = // ger skärmbredd i antal tecken
70      sys.env.get("COLUMNS").flatMap(_.toIntOption).getOrElse(default)
71
72    def moveCursor(row: Int, col: Int): Unit = // nästa utskrift sker vid (row, pos)
73      print(s"${EscapeCode}[${row + 1};${col + 1}H")
74
75    def hideCursor(): Unit = print(s"${EscapeCode}[?25l")  // gömmer blinkande markör
76
77    def showCursor(): Unit = print(s"${EscapeCode}[?25h")  // visar blinkande markör
78  end Terminal
```

-

Uppgift 4. `KaraokeTerminal`. (37p)

```
package solution

import scala.compiletime.ops.double

object KaraokeTerminal:
  case class Tempo(beatsPerMin: Int):
    require(beatsPerMin > 0, "beatsPerMin must be more than zero")
    val secondsPerBeat: Double = 60.0 / beatsPerMin
    def waitBeat(): Unit = Tempo.waitSeconds(secondsPerBeat)

  object Tempo:
    def waitSeconds(seconds: Double): Unit = Thread.sleep((seconds * 1000).round)
    val Default = Tempo(120)
    val SettingPrefix = "tempo="
    val Separator = ';'

    def fromString(s: String): Option[Tempo] =
      if s.startsWith(Tempo.SettingPrefix) then
        val bpm = s.stripPrefix(Tempo.SettingPrefix)
          .toIntOption
          .getOrElse(Tempo.Default.beatsPerMin)
        if bpm > 0 then Some(Tempo(bpm)) else None
      else None
  end Tempo

  case class Lyrics(tempo: Tempo, lines: Vector[String]):
    def showLine(i: Int): String = lines(i).filterNot(_ == Tempo.Separator)

  object Lyrics:
    private def loadLines(path: String): Vector[String] =
      val s = io.Source.fromFile(path, enc = "UTF-8")
      try s.getLines().toVector finally s.close()

    def fromFile(path: String): Lyrics = //tot 7p
      var lines = loadLines(path)
      Tempo.fromString(lines.lift(0).getOrElse("")) match
        case None => Lyrics(Tempo.Default, lines)
        case Some(tempo) => Lyrics(tempo, lines.drop(1))
  end Lyrics

class KaraokeTerminal(override val nRows: Int)
extends Terminal(nRows, Terminal.width()):
  require(nRows >= 3, "nRows must be at least 3")
  import KaraokeTerminal.*

  def playLine(row: Int, lineWithBeats: String, tempo: Tempo): Unit = //tot 8p
    var col = 0  //1p
    for ch <- lineWithBeats do  //2p
      if ch == Tempo.Separator then   //1p
        tempo.waitBeat() //1p
```

```
51        else
52          highlight(row, col)  //2p
53          col += 1  //1p
54      //Alternativ lösning:
55      // val it = line.iterator
56      // var col = 0
57      // while it.hasNext do
58      //   val c = it.next
59      //   if c == Tempo.separator then
60      //     tempo.waitBeat()
61      //   else
62      //     highlight(row, col)
63      //     col += 1
64      // end while
65
66    def play(lyrics: Lyrics): Unit = //tot 22p
67      clear()  //1p
68      Terminal.hideCursor()  //1p
69      val midRow = nRows / 2  //1p
70      val initLines = lyrics.lines.take(midRow + 1)  //3p
71      for i <- initLines.indices do //1p
72        put(lyrics.showLine(i), midRow + i, 0, isHighlight = false) //4p
73      var current = 0 //1p
74      while current < lyrics.lines.length do //1p
75        playLine(midRow, lyrics.lines(current), lyrics.tempo)  //3p
76        scroll() //1p
77        current += 1 //1p
78        val nextBottomLine =
79          if current + nRows - midRow > lyrics.lines.length  //2p
80          then ""                                            //1p
81          else lyrics.showLine(current + nRows - midRow - 1) //2p
82        put(nextBottomLine, nRows - 1, isHighlight =  false) //3p
83      end while
84      Terminal.showCursor() //1p
85      println() //1p
86  end KaraokeTerminal
```

-