Bedömningsmall för tentamen EDAA45 Programmering, grundkurs

2025-01-08, 8:00-13:00

Hjälpmedel: Snabbreferenser för Scala och Java.

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.
`u1`			`Set[Int]`	`Set(0)`
`u2`	**X**			
`u3`			`Set[Int]`	`Set(0, 1)`
`u4`	**X**			
`u5`			`Option[Int]`	`None`
`u6`			`Int`	`42`
`u7`			`Vector[Unit]`	`Vector(())`
`u8`		**X**		
`u9`	**X**			
`u10`			`Int`	`0`

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. Fyra-i-rad. Totalt max 80p.

Generella bedömningsriktlinjer 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.
- Felaktig användning av tom parameterlista (om den inte ska vara där eller om den 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.
- Deklaration av `val` när det måste vara `var` för att koden ska fungera 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.
- 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. `Marker`, `Pos`. (7p)

```
1  package solution
2
3  enum Marker:
4    case Empty, Cross, Ring
5    def toChar: Char = Marker.chars(ordinal)
6    def nonEmpty: Boolean = this != Empty
7
8  object Marker:
9    val chars: Seq[Char] = Vector(' ', 'X', 'O')
10   def fromChar(c: Char): Marker = //tot 3p
11     c match
12       case 'X' => Marker.Cross
13       case 'O' => Marker.Ring
14       case _ => Marker.Empty
15
16 enum Direction(val deltaRow: Int, val deltaCol: Int):
17   case UpDown    extends Direction(1, 0)
18   case LeftRight extends Direction(0, 1)
19   case Diag1     extends Direction(1, 1)
20   case Diag2     extends Direction(1, -1)
21
22 case class Pos(row: Int, col: Int):
23   def +(d: Direction): Pos = //tot 2p
24     Pos(row + d.deltaRow, col + d.deltaCol)
25   def -(d: Direction): Pos = //tot 2p
26     Pos(row - d.deltaRow, col - d.deltaCol)
```

- Med anledning av errata på tentan: I ursprungliga tentan förekommer det i texten `Mark` i stället för korrekta `Marker`. Inget avdrag ges om studenten i sin kod skrivit `Mark` i stället för `Marker`. Gäller alla uppgifter i hela tentan.

Uppgift 3. `Board`. (43p)

```scala
package solution

class Board:
  val (nbrRows, nbrCols) = (6, 7)

  private val board =
    import collection.{immutable, mutable}
    immutable.ArraySeq.fill(nbrCols)(mutable.ArrayBuffer.empty[Marker])

  def canPut(col: Int): Boolean = //tot 5p
    col >= 0 && col < nbrCols && board(col).length < nbrRows
    //eller:  board.lift(col).map(_.length < NbrRows).getOrElse(false)

  def availableCols: Seq[Int] = //tot 4p
    (0 until nbrCols).filter(canPut)

  def get(p: Pos): Marker = //tot 5p
    import p.*
    board.lift(col).map(_.lift(row).getOrElse(Marker.Empty)).getOrElse(Marker.Empty)

  def nextEmptyRow(col: Int): Option[Int] =  //tot 4p
    if canPut(col) then Some(board(col).length) else None

  def put(m: Marker, col: Int): Unit = //tot 4p
    require(canPut(col) && m != Marker.Empty)
    board(col).append(m)

  def countAdjacent(m: Marker, p: Pos, d: Direction): Int = //tot 10p
    var result = 0
    var current = p + d
    while get(current) == m do { result += 1; current += d }
    current = p - d
    while get(current) == m do { result += 1; current -= d }
    result

  def countAllAdjacentIfPut(m: Marker, col: Int): Option[Seq[Int]] = //tot 7p
    for row <- nextEmptyRow(col) yield
      Direction.values.map(dir => countAdjacent(m, Pos(row, col), dir)).toSeq

  def isWinnerIfPut(m: Marker, col: Int): Boolean = //tot 4p
    if !canPut(col) then false else
      countAllAdjacentIfPut(m, col).map(_.max >= 3).getOrElse(false)

  def show =
    val sepLine = ("-" * nbrCols) + "\n"
    val rowStrings = for r <- nbrRows - 1 to 0 by -1 yield
      (for c <- 0 until nbrCols yield get(Pos(r, c)).toChar).mkString
    sepLine + rowStrings.mkString("\n") + "\n" +
      sepLine + (0 until nbrCols).mkString
```

Uppgift 4. `Player`. (30p)

```scala
package solution

trait Player:
  def marker: Marker
  def nextCol(board: Board): Option[Int]

class HumanPlayer(val marker: Marker) extends Player:
  override def toString = s"Human Player ${marker.toChar}"
  def nextCol(board: Board): Option[Int] = //tot 12p
    val cols = board.availableCols
    if cols.isEmpty then None else
      val prompt = s"select one column of ${cols.mkString(",")}: "
      def input() = io.StdIn.readLine(prompt)
      var col: Option[Int] = None
      while
        col = input().toIntOption
        col.isEmpty || !board.canPut(col.get)
      do println("Illegal column. Try again!")
      col

class GreedyComputerPlayer(val marker: Marker) extends Player:
  override def toString = s"Greedy Computer Player ${marker.toChar}"
  def nextCol(board: Board): Option[Int] = //tot 18p
    val cols = board.availableCols
    if cols.isEmpty then None else
      val bestCounts: Seq[Int] =
        cols.map: c =>
          board.countAllAdjacentIfPut(marker, c).getOrElse(Seq(0)).max
      val best = bestCounts.max
      val cix = bestCounts.zipWithIndex.filter((c,i) => c == best)
      val iOpt = cix.lift(util.Random.nextInt(cix.length)).map(_._2)
      iOpt.map(i => cols(i))
```

- Med anledning av errata på tentan: I ursprungliga tentan var kolumnerna i exempelkörningen på sidan 5 ej kommaseparerad även om det står i uppgiften för `HumanPlayer.nextCol` att det ska vara så. Det blir inget avdrag för listan med kolumner i prompten oavsett om studenten använder något av `cols.mkString(",")` eller `cols.mkString` i prompten, så länge alla kolumner är med i prompten oavsett ev. separator dem emellan.