Bedömningsmall för kontrollskrivning EDAA45 Programmering, grundkurs

2024-10-29, 14:00-19:00

Hjälpmedel: Snabbreferenser för Scala och Java.

Instruktioner för kamraträttning

- Läs igenom hela denna bedömningsmall innan bedömningen påbörjas. Sätt dig in i mönsterlösningens angreppssätt och funktion, samt hur poängsättningen ska ske.
- För varje del: läs noga lösningen du ska bedöma och sätt dig in i det specifika angreppssättet.
- Ge maxpoäng om koden är korrekt, läsbar och lika bra som mönsterlösningen även om den är annorlunda. Det finns många olika sätt att lösa varje del, varav många är lika bra som mönsterlösningen. En lösning kan vara lika bra som mönsterlösningen, även om den är längre eller kortare eller har ett annat angreppssätt.
- Ge noll poäng om lösningen saknas, är oläsbar, obegriplig eller helt fel.
- Markering av poängavdrag (negativa) och uppgiftspoäng (inringad, positiv):
  - Markeringar ska göras med en penna med **avvikande färg**, gärna *röd*.
  - *Stryk under* de delar av en rad som är felaktiga eller bara delvis korrekta.
  - Om något saknas, *markera med en pil* var det som saknas borde finnas och skriv en kommentar om vad som saknas.
  - Markera poängavdrag i *högerkanten* med *minuspoäng* enligt bedömningsriktlinjerna i detta häfte.
  - När du markerat relevanta avdrag för en hel uppgift, beräkna och skriv total uppgiftspoäng *inringad* i en cirkel, så att det går lätt att skilja uppgiftspoäng från poängavdrag. Uppgiftspoängen ska inte vara negativ utan ligga i intervallet [*0, Maxpoäng*].
  - Om det efter studier av de markerade poängavdrag du gjort i skrivningen och dessa riktlinjer, inte är uppenbart varför du ej givit maxpoäng skriv då en kort förklaring i kanten.
  - Om en lösning är oläslig eller helt obegriplig, skriv *”oläsligt”* eller *”obegripligt”* i kanten.
- Avsluta med att göra en *kvalitativ helhetsbedömning* av hela skrivningen och justera poäng för vissa delar om du tycker att poängsumman för hela uppgiften inte överensstämmer med din helhetsbedömning. Om du ändrar i din poängsättning, stryk över de gamla poängen med ett kryss och skriv nya poäng bredvid.
- Summera poängen och fyll i summan på omslagets framsida.
- Ange era identitetsnummer i fälten för rättare på omslagets framsida.
- Kamraträttningar med invändningar kommer att bedömas igen i efterhand av lärare. Även de kamratbedömningar som godkänns utan invändningar kommer att stickprovsbedömas i efterhand av lärare.

Del A. Tolka uttryck. Totalt max 10 p.

Efter init. av:	Vid kompileringsfel sätt kryss.	Vid exekveringsfel sätt kryss.	Ange statisk **typ** som kompilatorn härleder om ej kompilerings- eller körtidsfel.	Ange det **värde** som tilldelas vid exekvering, med samma format som vid utskrift av värdets `toString`, om ej kompilerings- eller körtidsfel.
`u1`	**X**			
`u2`			`Char`	`g`
`u3`			`String`	`xkr`
`u4`		**X**		
`u5`			`Option[Char]`	`Some(g)`

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 ges inget avdrag, till exempel `10` i stället för `10.0` för ett värde av `Double`-typ, med eller utan citationsteckenpar `" "` runt en sträng eller `' '` runt ett tecken.
- Om typen är `Seq[T]` eller `Option[T]` så är det även OK att ange en mer specifik typ, t.ex. `Vector[T]` eller `Some[T]` eller `None` på typen *om* den mer specifika typen stämmer med värdet.
- Om rätt svar är ett kryss i någon av kolumnerna för kompilerings- eller exekveringsfel, men man kryssat för inkorrekt typ av fel, ges -1p i avdrag.
- Om man helgarderat med två kryss på samma rad ges -2p i avdrag.
- Om man svarat med fel typ men rätt värde ges -1p i avdrag.
- Om man svarat med rätt typ men fel värde ges -1p i avdrag.
- Om både värde och typ är fel ges -2p i avdrag.
- Om man svarat med typ/värde *och* kryss i någon av kolumnerna för kompilerings- eller exekveringsfel på samma rad ges -2p i avdrag.

Del B. Skriva kod. Totalt max 40 p.

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 -1p i avdrag.
- Felaktigt avskriven metodsignatur från specifikationen ger -1p i avdrag.
- Om nödvändigt semikolon saknas (t.ex. mellan flera satser på samma rad) ges -1p i avdrag.
- Felaktig användning av tom parameterlista (om den inte ska vara där eller om den saknas) ger -1p i avdrag.
- Mindre fel i likhet med föregående två punkter där det framgår tydligt vad som egentligen avses ger -1p i avdrag.
- Deklaration av `val` när det måste vara `var` för att koden ska fungera ger -2p i avdrag.
- Deklaration av `var` när det skulle kunna vara en `val` och fungera lika bra ger -1p i avdrag.
- Onödigt krångliga booleska uttryck, t.ex. `quit = if uttryck then true else false` i stället för bara `quit = uttryck`, ger -1p i avdrag.
- Onödiga typannoteringar ger inget avdrag.
- Tillägg av element i samling på fel sätt, t.ex. `+:` i stället för `:+` eller liknande, ger -1p i avdrag.
- Användning av `return` ger -2p i avdrag.
- Användning av metoden `length` på samling som ej är en sekvens, t.ex `Set` el. `Map`, ger -1p i avdrag.
- Enkla misstag som är lätta att åtgärda ger -1 eller -2p 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.
  Ett enda avdrag för *många* förekomster av samma fel kan göras för en mängd medlemmar, en hel uppgift eller till och med för hela skrivningen, **om felet är trivialt**, till exempel vid systematiska mindre syntaxfel så som flera onödiga semikolon vid radslut, många glömda slutmarkeringar: `} )` eller `then` eller `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 -1p 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 -2p i avdrag.

Lösningar och specifika bedömningsriktlinjer del B

- Mönsterlösningen nedan är bara ett av flera möjliga sätt att lösa uppgiften som är lika bra. Du ska vid bedömningen avgöra om den inlämnade lösningen är likvärdig mönsterlösningen eller om avdrag ska göras.
- Endast de poängmarkerade delarna t.ex. `//2p` ingår i bedömningen.
- Maxpoängen färdelas jämt över rader och (del)uttryck som ingår i nedan lösning, om inget annat anges i efterföljande punkter.
- Om motsvarande funktionalitet **helt saknas** eller det finns **stort fel** i motsvarande funktionalitet så dras motsvarande delpoäng, annars om det är ett **litet fel** så dras lämplig andel av delpoängen.
- Raden `package solution` ska ej ingå i studentens lösning.

Uppgift `Mark`

```
1  package solution
2
3  enum Mark:
4    case Empty, Absent, Incomplete, Approved
5    def toChar: Char = Mark.chars(ordinal)
6
7  object Mark:
8    val chars: Seq[Char] = Vector('.', '-', '+', '*')
9
10   def fromChar(c: Char): Mark = //tot 6p
11     val i = chars.indexOf(c)                 //3p
12     if i < 0 then Mark.Empty else values(i)  //3p
13 end Mark
```

Specifika bedömningsriktlinjer för denna uppgift:

- `fromChar`
  - Det går lika bra att göra linjärsökning med `find` och hantera defaultvärde med `getOrElse`:

    ```
    val i = chars.find(_ == c).getOrElse(0)
    values(i)
    ```

  - Helt felaktig eller avsaknad av linjärsökning ger -3p i avdrag.
  - Felaktig hantering av negativt index eller fel i villkor tex. `i <= 0` ger -1p i avdrag.
  - Avsaknad eller felaktig hantering av `Mark.Empty` ger -1p i avdrag.
  - Felaktig indexering i `values` ger -1p i avdrag.

Uppgift `Student`

```scala
package solution

case class Student(name: (String, String), id: String, marks: Seq[Mark]):
  def isApproved(labIndex: Int): Boolean = //tot 4p
    labIndex >= 0 && labIndex < marks.length && marks(labIndex) == Mark.Approved

  def isAllApproved(toWeek: Int): Boolean = //tot 6p
    marks.indices.take(toWeek).forall(isApproved)

  def count(m: Mark): Int = //tot 4p
    marks.count(_ == m)

  def showName(lastNameFirst: Boolean = true): String = //tot 3p
    if lastNameFirst then s"${name._2}, ${name._1}"
    else s"${name._1} ${name._2}"

  def showStudent(lastNameFirst: Boolean = true): String =
    val ms = marks.map(_.toChar).mkString
    s"$id ${count(Mark.Approved)} $ms ${showName(lastNameFirst)}"
```

Specifika bedömningsriktlinjer för denna uppgift:

- `isApproved`
  - Felaktig indexkontroll ger upp till -2p i avdrag.
  - Felaktig indexering i `marks` ger -1p i avdrag.
  - Felaktigt jämförelsevärde annat än `Mark.Approved` ger -1p i avdrag.
- `isAllApproved`
  - Algoritmfel där kontroll sker av alla även efter `toWeek` ger -3p i avdrag.
  - Det går lika bra att iterera med **`for`** eller **`while`**, men mellanresultat-variabler behövs då etc.
- `count`
  - Avsaknad av anrop av `marks.count` (eller motsvarande funktionalitet) ger -2p i avdrag.
  - Felaktig jämförelse som argument till `marks.count` (eller motsvarande funktionalitet) ger -2p i avdrag.
  - Det går också bra att räkna från grunden med **`for`** eller **`while`**, men mer komplicerat och större risk att göra fel.
- `showName`
  - Avsaknad av kontroll av `lastNameFirst` ger $--$ 2p i avdrag.
  - Felaktig visning, tex. saknat kommatecken, vid efternamn först ger $--$ 1p i avdrag.
  - Felaktig visning, tex. kommatecken, vid förnamn först ger $--$ 1p i avdrag.

Uppgift `Register`

```scala
package solution

case class Register(students: Vector[Student]):
  def countAllApproved(toWeek: Int): Int = students.count(_.isAllApproved(toWeek))

object Register:
  val NbrOfLabs = 7

  def fromStringToMarks(s: String): Seq[Mark] = //tot 5p
    for i <- 0 until NbrOfLabs yield //1p
      if i < s.length                //1p
      then Mark.fromChar(s(i))       //2p
      else Mark.Empty                //1p
    /*  eller t.ex.:
    s.take(NbrOfLabs).map(Mark.fromChar).padTo(NbrOfLabs, Mark.Empty)
    */

  def fromLineToStudent(line: String, delim: String): Student = //tot 12p
    val xs = line.split(delim)                                              //1p
    require(xs.length >= 3 && xs.length <= 4, "felaktigt antal kolumner") //2p
    require(xs.take(3).forall(_.nonEmpty), "tomma kolumner")               //3p
    val ms =
      if xs.length == 4                                                     //1p
      then fromStringToMarks(xs(3))                                         //1p
      else Seq.fill(NbrOfLabs)(Mark.Empty)                                  //2p
    Student(name = (xs(0), xs(1)), id = xs(2), marks = ms)                  //2p

  def fromFile(fileName: String): Register =
    val lines = scala.io.Source.fromFile(fileName, "UTF-8").getLines.toVector
    Register(lines.drop(1).map(line => fromLineToStudent(line, ",")))
end Register
```

Specifika bedömningsriktlinjer för denna uppgift:

- `fromStringToMarks`
  - Iterering över felaktiga gränser ger -1p i avdrag.
  - Felaktig hantering av `Mark.Empty` t.ex. vid överskjutande tecken i `s` ger -2p i avdrag.
  - Felaktig hantering av `Mark.fromChar` ger -2p i avdrag.
- `fromLineToStudent`
  - Felaktig användning av split ger -1p i avdrag.
  - Felaktigt villkor vid test av fel antal kolumner ger -2p i avdrag.
  - Felaktig villkor vid test av tomma kolumner ger -3p i avdrag.
  - Missad kontroll av 3 eller 4 kolumner ger -1p i avdrag.
  - Felaktigt anrop av fromStringToMarks ger -1p i avdrag.
  - Felaktig sekvens med tomma markeringar vid 3 kolumner ger -2p i avdrag.
  - Felaktiga klassparametrar vid konstruktion av student ger -2p i avdrag.