Tentamen
Nätverksprogrammering
Del 2

2018–08–23, 8.00–13.00

Tillåtna hjälpmedel för denna del av tentamen:

- Java snabbreferens.
- Kursboken: Java Network Programming av Eliotte Rusty Harold.
- Valfri lärobok i Java.
- Utskrift av OH-bilder från föreläsningarna.

Denna tentamen i kursen Nätverksprogrammering består av två delar – en del som innehåller frågor av teoretisk/principiell/utredande karaktär och en del som innehåller praktiska programmeringsuppgifter. Detta är del 2. Den ska du ha erhållit tillsammans med ett färgat tentamensomslag när du lämnade in din lösning på del 1 av tentamen.

För godkänt betyg på tentamen krävs sammanlagt minst 20 poäng på tentamen, varav minst 8 poäng på vardera deltentamen.

---

1. Resultatpresentation för programmeringstävling

Institutionen för Datavetenskap vid LTH arrangerar årligen tillsammans med flera andra nordiska universitet NM i programmering för högskolestuderande. NM-tävlingen är en distribuerad tävling som genomförs parallellt vid ett antal högskolor i Sverige, Norge, Danmark, Finland och på Island. Tävlingen fungerar även som lokal uttagningstävling till europafinalen i en världsomfattande programmeringstävling organiserad av Association of Computing Machinery (ACM). De bästa från europafinalen går sedan vidare till världsfinal på något exotiskt resmål i världen. Tävlingen går ut på att ett lag bestående av tre studenter under fem timmar och med en dator skall lösa så många programmeringsproblem som möjligt på så kort tid som möjligt. Den som har löst flest uppgifter vid tävlingens slut vinner. Om två lag har löst lika många uppgifter vinner det lag som löst sina uppgifter på kortast tid: Varje gång en korrekt lösning lämnas in tas aktuell tidpunkt mätt i minuter sedan tävlingsstarten och läggs till lagets totaltid. Ju lägre totaltid desto bättre.

Hjälp oss genom att skriva en enkel resultatserver som håller reda på aktuell ställning mellan lagen enligt nedanstående instruktioner:

- De lokala tävlingsledarna kopplar upp sig mot resultatservern med hjälp av Telnet.
- En tävlingsledare kan göra tre saker:
  1. Rapportera in en ny ställning för ett lag, till exempel när de har löst ytterligare en uppgift.
  2. Begära en utskrift av aktuell ställning.
  3. Gå in i *resultatstationsläge* (se beskrivning nedan).
- För att ange en ny ställning för ett lag skriver tävlingsledaren lagnummer, totalt antal lösta uppgifter och ny totaltid på en rad och trycker på Enter.

  Exempel: `6 3 231`

  Om lagnumret (6 i exemplet) inte förekommit tidigare läggs ett nytt lag med det numret in i resultatlistan. I annat fall uppdateras bara de gamla uppgifterna om laget.

- För att begära en utskrift av aktuell ställning skriver tävlingsledaren ”list” ensamt på en rad och trycker på Enter, varvid servern svarar med en lista över aktuell ställning. Listan är sorterad med bästa laget först och sämsta laget sist med ett lag på varje rad och uppgifter om lag, antal lösta problem och totaltid.
- Telnetförbindelsen ska hela tiden förbli öppen så att tävlingledaren snabbt kan ge nya kommandon till resultatservern.
- Servern ska kunna klara godtyckligt (men naturligtvis rimligt) antal samtidiga uppkopplingar av tävlingsledare/resultatstationer.
- En *resultatstation* är en dator som står ute bland de tävlande och som hela tiden visar en resultatlista med den aktuella ställningen. Före tävlingen startar tävlingsfunktionärerna Telnet på dessa datorer, ansluter till resultatservern och ger kommandot ”station” för att gå in i resultatstationsläge. Detta innebär att den aktuella ställningen skrivs ut. När ställningen sedan ändras skrivs en ny resultatlista ut *automatiskt*. När man väl gått över i resultatstationsläge finns det inget sätt att avbryta utan att stänga ner förbindelsen till servern.
- För att hålla reda på alla resultaten behövs någon form av enkel databas. Då skrivandet av koden för en dylik visserligen kan vara intressant som programmeringsuppgift men kanske av föga intresse ur nätverkssynvinkel ges nedan en specifikation av en färdig klass som du kan använda för att lagra och skriva ut resultaten:

  ```
  class Results {
      /** lägger in ett uppgifter om ett nytt lag i databasen eller
          uppdaterar tidigare uppgifter om laget.
          Returnerar true om databasen ändras,
          false om samma uppgifter redan fanns i databasen */
      public boolean newResult(int team, int problems, int time);

      /** skriver ut en resultatlista sorterad efter lagens aktuella
          placering på den OutputStream som anges i parametern */
      public void printResults(OutputStream os);
  }
  ```

  Tänk dock på att oväntade saker kan hända, t.ex. att databasen blir korrupt, ifall två trådar samtidigt exekverar någon av operationerna i klassen. Skriv ditt program på ett sådant sätt att detta garanterat inte kan förekomma.
- Servern ska vara rimligt robust så att den överlever om en förbindelse till en tävlingsledare kopplas ned oväntat.
- För enkelhetens skull kan du förutsätta att användarna alltid skriver korrekta kommandon.
- `String.split("\s")` returnerar alla ord i strängen i en vektor, t.ex `"6 3 231".split("\s")` returnerar `["6", "3", "231"]`.
- När man trycker på Enter i Telnet skickar programmet två tecken över nätverksförbindelsen. Dessa är *carriage return* (CR, ’\r’, ASCII-kod 13) följt av *line feed* (LF, ’\n’, ASCII-kod 10). I övrigt sänds enskilda tecken som 8-bitars ASCII-tecken.
- Observera att olika implementationer av Telnet uppför sig lite olika när det gäller exakt när tecken skrivs till nätverksförbindelsen. Om man kör Telnet på en Unix-baserad dator kommer Telnet att vänta tills man trycker på Enter och då skriva hela raden följt av radslutstecknen. På en Windows-dator däremot kommer tecken att skickas så fort man trycker ner en tangent (alltså som enskilda tecken). För ett korrekt skrivet serverprogram (som ger maximal tentamenspoäng) kommer detta dock inte ha någon betydelse.

(20p)

Slut!