Tentamen
Nätverksprogrammering
Del 1

2017–06–02, 8.00–13.00

Tillåtna hjälpmedel för denna del av tentamen: *inga*. Kurslitteratur och andra hjälpmedel för del 2 av tentamen skall förvaras på golvet bredvid bordet eller vid salens vägg.

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 1. När du löst uppgifterna i denna del av tentamen lämnar du in din lösning i det vita tentamensomslaget varvid du erhåller del 2 av tentamen tillsammans med ett nytt, färgat, tentamensomslag som skall användas vid inlämning av din lösning på del 2 av tentamen.

För godkänt betyg på tentamen krävs sammanlagt minst 20 poäng på tentamen, varav minst 8 poäng på vardera deltentamen. För högre betyg krävs mer, så gör så många uppgifter du kan.

---

1. För att hålla reda på trådars tillstånd i en monitorer finns mängderna *entry set* och *waiting set*.
   a) Förklarar kortfattat syftet med de två mängderna.

   (2p)

   b) När läggs trådar till, respektive tas bort från mängderna?

   (1p)

2. HTTP är ett populärt protokoll för kommunikation på internet.
   a) Förklarar kortfattat kommandona GET, PUT och POST. Du behöver inte ange exakt syntax.

   (2p)

   b) Hur vet man att en HTTP operation lyckas? Om operationen misslyckas, kan man då få någon form av felmeddelande?

   (1p)

3. JavaScript använder som programmeringsspråk för dynamiska hemsidor.
   a) JavaScript använder *function scope*. Förklara kortfattat vad det innebär.

   (1p)

   b) Förklara kortfattat *closure*. Ge ett exempel.

   (2p)

4. Vilka av följande sex påståenden är korrekta? Rätt svar get *+0,5p*, felaktigt svar *-1p*. Lämnas inget svar ger det *0p*. Hela uppgiften kan inte ge en negativt poäng.
   a) För att upprätta en TCP-förbindelse till ett program som kör på en annan dator i nätverket räcker det med att vi känner till den andra datorns IP-nummer.
   b) Portnummer 1-1023 är normalt reserverade för TCP-trafik.
   c) En socket representerar i TCP-sammanhang en enkelriktad förbindelse över nätverket.

---

d) Om ett program lyssnar efter UDP-paket på ett visst portnummer kan ett annat program på samma dator starta en TCP-server som accepterar TCP-uppkopplingar på samma portnummer.

e) När vi skickar data via en TCP-förbindelse delas vår data automatiskt upp i mindre bitar som skickas i form av UDP-paket. När UDP-paketen tas emot sätts datan i UDP-paketen automatiskt ihop igen till en kontinuerlig dataström.

f) UDP-protokollet garanterar att innehållet i de datapaket som anländer till mottagaren inte har förvanskats på vägen p.g.a. nätverksfel.

(3p)

5. Hemsidor byggs normalt upp av flera filer.

a) Vilken information finns i HTML och CSS-filerna?

(2p)

b) Vad är DOM i kontexten av en hemsida?

(1p)

c) Hur kan man från JavaScript modifiera utseendet på en hemsida?

(1p)

6. Betrakta koden nedan. Önskar resultat är utskriften av alla positiva heltal, *1 2 3* ..., ett tal per rad. Ordningen har ingen betydelse, men alla heltal ska skrivas ut. Resultatet blir dock inte det önskade. Varför? Använd vedertagna begrepp för att beskriva felen. Vad behöver ändras för att få det önskade resultatet?
tips: Vad händer om programmet körs i en miljö som bygger på samarbetande schemaläggning (cooperative scheduler)?

(4p)

```
public class Tenta2017 {

    public static void main(String[] args) {
        Monitor m = new Monitor();
        Producer p1 = new Producer(m, 1);
        Producer p2 = new Producer(m, 2);
        Consumer c1 = new Consumer(m);
        c1.start();
        p1.start();
        p2.start();
    }
}
```

Listningen fortsätter på nästa sida

```
public class Monitor {
    private int n;

    synchronized int get() {
        int tmp = n;
        n = 0;
        return tmp;
    }

    synchronized void put(int n) {
        this.n = n;
    }

}

public class Producer extends Thread {

   private Monitor m;
   int n;

   Producer(Monitor m, int n){
       this.m = m;
       this.n = n;
   }

    void run() {
        while(true){
            m.put(n);
            n = n + 2;
        }
    }

}

public class Consumer extends Thread {

   private Monitor m;

   Consumer(Monitor m){
       this.m = m;
   }

    void run() {
        while(true){
            int n = m.get();
            if(n > 0){
                System.out.println(n);
            }
        }
    }

}
```

Slut på del 1 – lämna in och hämta ut del 2!