LULEÅ TEKNISKA UNIVERSITET

# TENTAMEN

| **Kursnamn:** Nätverksprogrammering | **Kurskod:** **D0036D** |
|---|---|
| **Antal timmar:** 5 | **Datum:** 20191220 |

| **Tillåtna hjälpmedel:** Inga |
|---|

| **Jourhavande lärare m fullständigt telefonnr:** **Örjan Tjernström 0910-585716** | **Jourhavande lärare m fullständigt telefonnr:** |
|---|---|
| **Jourhavande lärare m fullständigt telefonnr:** | **Jourhavande lärare m fullständigt telefonnr:** |

| **Betygsgränser:** 50% betyg 3. Gränser för betyg 4 och 5 bestämmes senare. | **Antal uppgifter och totalpoäng:** 10 40 |
|---|---|

**Övriga uppgifter:**

Elektronisk utrustning skall vara avstängd under tentamen.

Svaren på nedanstående uppgifter skall vara relaterade till aktuell kurs för att kunna ge poäng.

**Allmänna anvisningar**
Kontrollera att du fått samtliga uppgifter.
Besvara endast en uppgift per lösningsblad. Skriv inte på baksidan.
Skriv tydligt, texta gärna och använd inte röd penna.

**Efter tentamen**
Tentamensresultat meddelas senast tre veckor efter tentamenstillfället och senast två veckor före nästa omtentamenstillfälle.
Tentamensresultatet syns på Mitt LTU – Ladok för studenter.
Din rättade tentamen skannas och blir synlig på Mitt LTU – Rättade tentor.

**Uppgifter till tryckeriet**

| **Projektnummer:** 341980 | **Antal exemplar:** 11 | **Antal sidor:** |
|---|---|---|
| **Övriga upplysningar:** | | |

Dina svar ska vara relaterade till kursen och endast gå att tolka på ett sätt. Endast översta frågan besvarad på ett svarsblad rättas så besvara max en fråga per svarsblad.

1. **Molntjänster:**
   a. Ni har fått ett uppdrag att utveckla en nätverksbaserad tjänst som för varje användare tar emot en fil, krypterar den och skickar en krypterad version tillbaka till användaren. Beskriv en arkitektur av tjänsten som är anpassat för att skaleras/skalas ut. (2p)
   b. Beskriv kortfattat funktionalitet av en lastbalanserare. (2p)

2. **Nätverk och spel:**(3p)
   a. Vad är dead reckoning i sammanhang med spel och nätverk?
   b. Vilken typ av information/data skulle man använda för att implementera dead reckoning i ett biltävlingsspel?
   c. Vilket problem kan uppstå när man använder sig av dead reckoning och beskriv två metoder för att lösa det?

3. **Nätverk och spel:** Massively multiplayer online (MMO) behöver använda sig av ett nätverk av Servrar (Server-network) för att dela arbetsbördan. Två angreppssätt för uppdelningen har belysts i kursen. (4p)
   a. Beskriva dessa.
   b. Vilka för och nackdelar har dessa två?

4. **Sockets**: Berkeley socket API innehåller en rad funktioner, vissa används till TCP andra till UDP och några till både TCP och UDP. Vilka funktioner, från Berkeley socket API, används och i vilken ordning används de om du ska implementera en
   a. TCP-client? (3p)
   b. UDP-server? (3p)

5. **Serialisering:**
   a. Vad innebär Serialisering?(1p)
   b. Vad gör seariliseringen så mycket lättare i Java jämfört med C eller C++?(1p)

6. **Multi Client Server**:
   a. Du ska implementera en TCP server som ska betjäna flera anslutna clienter samtidigt. Beskriv hur du löser detta. Använd Sekvensdiagram, kod eller liknande i din beskrivning.(4p)

7. **Trådar:**
   a. Beskriv begreppet race conditions.(1p)
   b. Beskriv begreppet data race.(1p)

8. **Endianes:** Host1 skickar en struct, bestående av 3 st 32-bit unsigned integers, till Host2. Host1 körs på en maskin som använder byteordningen Little endian. Host2 körs på en maskin som använder byteordningen Big endian. De viktiga delarna av koden för Host1 och Host2 finns i Bilaga 1.
   a. Om structens värden hos Host1 är x=4 278 190 080, y=268 505 089, z=16 000 000, vilka värden kommer då Host2 att ha i sin struct efter att överföringen slutförts? Svara med decimala eller hexadecimala värden. (se Tabell 1 i Bilaga 1) (2p)
   b. Åtgärda i koden så att värdena hos Host2 överensstämmer med de som Host1 överför. (2p)

9. **URL:**
   a. Vilka delar består en URL av? (1p)

10. **Applikationsprotokoll:** Du har implementerat en spelplan som en server. Spelplanen består av på 101*101 rutor. Rutorna kan ha tjugo färgnyanser eller vara utan färg. Din spelplan skall vara tillgänglig för flera clienter. UDP används för att kommunicera med din spelplan. Spelplanen skall hantera upp till och med tjugo samtidiga clienter. Varje client har sin egen färg. Clienterna har rätt att antingen skriva sin färg till en tom ruta eller radera sin egen färg från en ruta. Din spelplan kontrollerar att detta följs. Det är av största vikt att klienterna får veta om deras kommando kunde utföras eller inte. För att kommunicera med din spelplan behövs ett protokoll. I framtiden kan nya versioner av protokollet komma att skapas och spelplanen ska då kunna hantera alla versioner. Din uppgift är att konstruera ett protokoll som klarar detta. Ett antal meddelanden måste definieras i protokollet. Meddelandena skall vara i binärt format (ej klartext). Protokollet skall vara språkneutralt. Fält som förekommer i alla meddelanden kan ses som en header. Beskriv så att en, för dig okänd, programmerare kan implementera ditt protokoll. Till din hjälp har du nedanstående frågor.(10p)
    a. Vilka meddelanden har ditt protokoll?
    b. Vilka syften har protokollets meddelanden.
    c. Beskriv hur headern ser ut?
    d. Beskriv hur resten av respektive meddelande ser ut?

# Bilaga 1

| Decimal representation | Hexadecimal representation |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 65 792 | 1 0100 |
| 16 000 000 | F4 2400 |
| 16 776 960 | FF FF00 |
| 268 505 089 | 1001 1001 |
| 4 278 190 080 | FF00 0000 |
| 4 294 967 295 | FFFF FFFF |

**Tabell 1**

```
/*
Struct Test finns hos både Host1 och Host2.
*/

struct Test
{
            unsigned int x;
            unsigned int y;
            unsigned int z;
};


            /*
            Host1 kod
            */
            Test test;

            /*
            För uppgiften oviktig kod bortkommenterad.
            */

            int size = sizeof Test;
            char* buff = new char[size];
            memcpy((void*)buff, (void*)&test, sizeof Test);

            /*
            Här är koden, som ska skicka
            innehållet i buff till Host2, bortkommenterad.
            */

            /*
            Host1 kod slut.
            */

            /*
            Host2 kod
            */

            Test test;
            int size = sizeof Test;
            char* buff = new char[size];

            /*
            Här är koden, som tar emot trafik från Host1 och lägger
innehållet i buff, bortkommenterad.
            */

            memcpy((void*)&test, (void*)buff, sizeof Test);

            /*
            Host2 kod slut.
            */
```