Q1. Circuit switching in the Internet
In the Internet backbone, packet-switching is employed as the switching mechanism instead of circuit-switching. Circuit switching however was used before the Internet in the early telephone networks.

Q1.1: Name at least 2 reasons why employing circuit switching is not a good idea for the Internet in general.

Possible reasons:

- High latency to establish an end-to-end circuit, which is especially harmful to short flows;
- The workload of users within the Internet are bursty, and thus pre-allocating bandwidth would lead to under-utilization as it takes time before the circuit is freed again;
- The Internet routers and switches can fail, and it takes time before these failures are detected and propagated to all ISPs to renegotiate the circuit. In packet switching only the local ISP in which the fault occurs has to reroute packets around the fault;
- Circuit switching forces massive amount of circuit state to be kept in the switches. Secondly, additional complexity and scheduling is needed to enforce the circuit-allocated bandwidth.

Q1.2: Despite all the downsides you answered in Q1.1, can you give an example of an application which would greatly benefit from a circuit-switched Internet over the current packet-switched Internet? Why?

Circuit switching does not need large buffers because application rates are fixed. Without buffers the latency of packets is very low. Applications such as gaming,
tele-surgery or Skype calls have a traffic flow which is long-lasting, and their quality of experience (QoE) is heavily dependent on response time. Consistently guaranteed low round trip time would improve their functioning substantially, which is not anywhere near-optimal in the packet-switched Internet.

Q2: How would you implement flow prioritization in circuit switching? And in packet switching?

Circuit switching already provides the ability to assign more bandwidth to particular flows. Packet switching requires a priority field in the packet header, such that the switch can queue the packets based on their priority.

Q3: The success of cloud services is primarily thanks to the principle of resource sharing. Explain this with regards to network connectivity from the perspective of a medium-sized company focused on selling toys. Name at least 2 practical reasons.

Possible reasons:

- **Lack of statistical multiplexing ability**: A toy company has certain periods of massive amounts of shoppers visiting their website, e.g., on Black Friday. If the company would have to build a network to handle this peak load (as they would lose revenue if they did not), the network would remain severely underutilized during the remainder of time.

- **Locality**: The company’s infrastructure would need good Internet access and manage content distribution on a world-wide scale. Though necessary for a toy company, maintaining and operating these data centers across the globe is highly specialized labor that a non-technical company does not possess.

- **Growth response**: To facilitate growth, a first expensive initial purchase of equipment has to be made. In cloud-services you can scale on-demand.

Q4. Provisioning

Assume that $N$ users share the same up-link to their Internet Service Provider. The link capacity is $C$ Mbps. When online, a user will require $U$ Mbps bandwidth. The probability of a user being online at any point in time is $P_{\text{online}}$ independent of other users.

Q4.1: In closed form, what is the probability $P_{\text{inadequate}}$ that there is not enough available link capacity to fully satisfy the needs of the online users?

Let’s say the amount of discrete users online is $X$. Because the probability of a user being online is independent from other users, $X$ follows a binomial distribution:

$$P(X = x) = \operatorname{Bin}(x; P_{\text{online}}, N)$$
The probability of being inadequate is when $X > \text{floor}(\frac{C}{U})$, i.e. when the demands of the users exceed the available link capacity. From this follows:

$$P_{\text{inadequate}} = P(X > \text{floor}(\frac{C}{U})) = \sum_{i=\text{floor}(\frac{C}{U})+1}^{N} \text{Bin}(i; P_{\text{online}}, N)$$

Q4.2: For $N = 5$ users, $C = 20$ Mbps, $U = 7$ Mbps, and $P_{\text{online}} = 0.3$: what is the probability $P_{\text{inadequate}}$ of the link being inadequate?

$$\text{floor}(\frac{C}{U}) = \text{floor}(\frac{20}{7}) = 2$$

$$P_{\text{inadequate}} = \sum_{i=3}^{5} \text{Bin}(i; 0.3, 5) = 0.16308$$

Q4.3: Because provisioning for peak usage is too expensive, provisioning is done by keeping risk within a reasonable bound. In this exercise, risk is the probability of the link being inadequate. Given $(N, U, P_{\text{online}}, P_{\text{max.risk}})$ we want to know: what is the minimum $C$ such that $P_{\text{inadequate}} \leq P_{\text{max.risk}}$? Write the algorithm that answers this question.

The problem statement is:

$$P_{\text{inadequate}} = \sum_{i=\text{floor}(\frac{C}{U})+1}^{N} \text{Bin}(i; P_{\text{online}}, N) \leq P_{\text{max.risk}}$$

Insights:

- Because of the floor and summation, there is no continuous analytic solution for $C$;
- $\text{Bin}(i, P_{\text{online}})$ is independent of the choice of $C$, and $0 \leq i \leq N$;
- $i = \text{floor}(\frac{C}{U}) + 1$, so we only need to test $C$ as multiples of $U$. Any capacity less than $U$ added to the multiple of $U$ will not allow us to have more users simultaneously online.

Algorithm:
function Min-Link-Capacity-Needed(\(N, U, P_{\text{online}}, P_{\text{max.risk}}\))
  \(\text{sum} = 0.0\)
  for \(i \in 0, 1, ..., N\) do
    \(\text{sum} = \text{sum} + \text{Bin}(i; P_{\text{online}}, N)\)
    if \(\text{sum} \geq 1 - P_{\text{max.risk}}\) then
      return \(i \times U\)
    end if
  end for
end function

Q4.4: Implement the algorithm of Q4.3 in a language of your choosing. For \(N = 130\) users, \(U = 4\) Mbps, and \(P_{\text{online}} = 0.2\): what is the minimum link capacity \(C_{\text{min}}\) needed such that \(P_{\text{inadequate}} \leq 0.01\)?

\(C_{\text{min}} = 148\)

Algorithm in Java:
```java
package com.exampleworkspace.xpt.networks2018;
import java.math.BigDecimal;
import java.math.BigInteger;
public class Exercise1 {
    public static void main(String args[]) {
        BigInteger N = BigInteger.valueOf(130);
        BigDecimal U = BigDecimal.valueOf(4);
        BigDecimal POnline = BigDecimal.valueOf(0.2);
        BigDecimal PMaxRisk = BigDecimal.valueOf(0.01);
        System.out.println(calculateCMin(N, U, POnline, PMaxRisk));
    }
    private static BigDecimal calculateCMin(BigInteger N, BigDecimal U,
                                            BigDecimal POnline, BigDecimal PMaxRisk) {
        BigDecimal sum = BigDecimal.ZERO;
        for (int i = 0; i < N.intValue(); i++) {
            sum = sum.add(binomialPMF(BigInteger.valueOf(i), N, POnline));
            if (sum.compareTo(BigDecimal.ONE.subtract(PMaxRisk)) > 0) {
                return U.multiply(BigDecimal.valueOf(i));
            }
        }
        return null;
    }
    private static BigDecimal binomialPMF(BigInteger kBI, BigInteger nBI, BigDecimal pBD) {
        return new BigDecimal(factorial(nBI)).divide(new BigDecimal(factorial(kBI)).multiply(factorial(nBI.subtract(kBI)))).multiply(pBD.pow(kBI.intValue())).multiply(BigDecimal.ONE.subtract(pBD).pow(nBI.subtract(kBI).intValue()));
    }
    private static BigInteger factorial(BigInteger i) {
        if (i.equals(BigInteger.ZERO)) {
            return BigInteger.ONE;
        } else {
            return i.multiply(factorial(i.subtract(BigInteger.ONE)));
        }
    }
}
```

Q4.5: Suppose now that users’ requested bandwidth \(U\) is no longer constant \textit{a priori}, but follows a \textit{Poisson}(\(\lambda\)) distribution. I.e. for any user \(i\), \(U_i \sim \text{Poisson}(\lambda)\). What is the probability \(P_{\text{inadequate}}\) now?

\textit{Hint: the distribution of the sum of independent Poisson distributions is also a Poisson distribution.}
With the original constant distribution, the following could be easily converted into a partial sum:

\[ P(\text{inadequate} \mid X = x) = \begin{cases} 1 & \text{if } x \times U > C \\ 0 & \text{else} \end{cases} \]

With \( U_i \sim \text{Poisson}(\lambda) \) it becomes the following:

1(134,645),(941,718)

A key insight is that the sum of the Poisson distributions is also a Poisson distribution. \( \sum_{i=1}^{x} \text{Poisson}(\lambda) = x \cdot \text{Poisson}(\lambda) = \text{Poisson}(\lambda x) \). So the sum distribution \( M \sim \text{Poisson}(\lambda x) \). From this follows:

\[ P(\text{inadequate} \mid X = x) = P(M > C) = 1 - P(M \leq C) = 1 - e^{-\lambda x} \sum_{i=0}^{\text{floor}(C)} \frac{(\lambda x)^i}{i!} \]

Now, we put this all together:

\[ P(\text{inadequate}) = * \sum_{x=0}^{N} P(\text{inadequate} \wedge X = x) \]

\[ = ** \sum_{x=0}^{N} P(X = x) \cdot P(\text{inadequate} \mid X = x) \]

\[ = \sum_{x=0}^{N} \text{Bin}(x; P_{\text{online}}, N) \cdot P(\text{inadequate} \mid X = x) \]

\[ = \sum_{x=0}^{N} \text{Bin}(x; P_{\text{online}}, N) \cdot (1 - e^{-\lambda x} \sum_{i=0}^{\text{floor}(C)} \frac{(\lambda x)^i}{i!}) \]

* Law of total probability
** Bayes’ theorem

Q5: DNS Statements

1. Every IP address maps to a host name. (true / false) false
2. A host name can map to multiple IP addresses. (true / false) true
3. No two host names can map to the same IP addresses. (true / false) false
4. There can be one or more physical servers behind an IP address. (true / false) true

**Q6:** The *hosts.txt* file still exists on systems (i.e. in */etc/hosts/*). Why is it only writable with administrator privileges? Similarly, why is it important that we only use trusted DNS servers?

Firstly, many applications rely on DNS to function, so if the DNS records are invalid these applications will not work properly. Secondly, when the DNS records are maliciously edited, an attacker can redirect your requests. For example, you go to *bank.com* in your browser and are redirected to a phishing copy hosted on an attacker’s server.