1-3 Virtual memory

In Inspera, questions 1-6 have 10 points. This must be ignored. We use 0-3 points for every question. As the exam cover page states, every question has the same value for the exam.

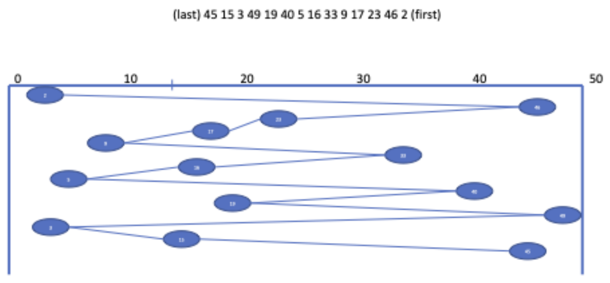
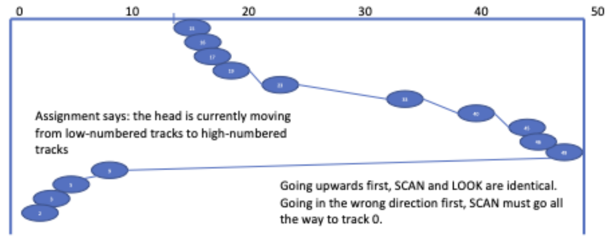
	Question in English	**Spørsmål på norsk**	**Retteveiledning**
1	What is virtual memory? Why do we use it and how does it work?	Hva er virtuelt minne? Hvorfor bruker vi det og hvordan fungerer det?	● 3: Virtual memory can exceed the physical RAM of a computer and use other storage resources as well. We want to use it to allow processes to use a large amount of memory without a lot of effort for the programmer. It requires hardware support (MMU) that can (a) translate transparently the addresses used by a running process from virtual address to physical address, (b) detect that a virtual address does not have a physical address right now (page fault), and start the page replacement procedure, (c) update the MMU data. ● 2: the explanation is missing either the answer to "Why do we use it" or one of the major points ● 1: only the basic idea is recognizable
2	Assume you have a 32-bit memory system, you are using 4 KB pages, and have a request for a particular virtual memory address. Explain how a traditional 1-level memory look-up works, and how the	Anta at du har et 32-bits minnesystem, du bruker 4 KB-sider og har en forespørsel om en bestemt virtuell minneadresse. Forklar hvordan et tradisjonelt 1-nivås minneoppslag fungerer, og hvordan den fysiske	● 3: You need 12 bits for the offset in the page, so the page table is 2^20 entries long (approx 1 mio). Each entry contains at least the base address in physical memory (if the address is in physical memory) or info about its location on

	physical address is found.	adressen blir funnet.	disk, a dirty bit and a present bit. If written to, dirty is set. If looked up and not present, start the replacement procedure. If the replacement procedure expunges a dirty page, write it to swap space on disk first. If looked up and the page is present, take the 20 bits from the page table entry and add the least significant bits from the requested virtual address, and you have the physical memory address. ● 2: One major part is missing, for example the writeback or the means for finding the actual physical address. ● 1: Rudimentary knowledge of the process can be seen in the answer.
3	When our memory is full, we need a way to replace elements so that we can fetch new requested data. Can you give an example of such an algorithm, and briefly discuss what properties such an algorithm should have?	Når minnet vårt er fullt, trenger vi en måte å erstatte elementer slik at vi kan hente nye forespurte data. Kan du gi et eksempel på en slik algoritme, og kort diskutere hvilke egenskaper en slik algoritme bør ha?	A page replacement algorithm should be simple (efficient, low overhead), make optimal decisions (read from and write to disk very rarely). A possible replacement algorithm is LRU, which reshuffles the list of pages in physical memory every time a read or write operation to a page is performed. When a page is addressed, it is moved to the top of the list. When a different page needs to be swapped in, the page that is at the tail end of the sorted list is replaced (ie. written to disk if dirty, and its physical memory overwritten by the requested page from disk). ● 3: According to the task, it is sufficient to write the algorithm’s name and the

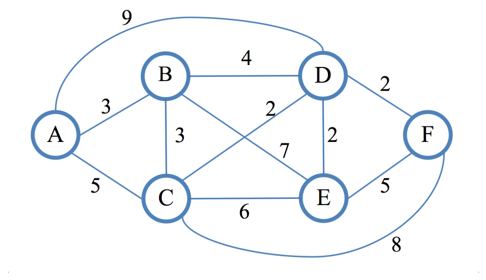
			properties “simple”+”good” decisions. The reason is that the task uses the word “such an algorithm” instead of “this algorithm”. ● 2: One of the points is correct. The name of the algorithm makes no sense (e.g. Round Robin makes no sense), or the properties are wrong. ● 1: Just one property is correct.

4-6 Storage

	Question in English	Spørsmål på norsk	Retteveiledning
4	Accessing secondary storage like mechanical hard drives takes a lot of time compared to accessing memory in RAM. Briefly explain why and what operations the access latency consists of.	Tilgang til sekundær lagring som mekaniske harddisker tar mye tid sammenlignet med tilgang til minne i RAM. Forklar kort hvorfor og hvilke operasjoner tilgangsforsinkelsen består av.	● 3: Mechanical hard drives must move the disk head to the correct position on the disk to read the requested data. This requires (a) motion of the disk arm, which is very slow, and (b) rotation of the disk, which is fairly slow. Optional: Once this position is reached, data must be read and copied into RAM. ● 2: e.g. one of the points above is mentioned, but not more. Possible some incorrect statements. ● 1: Just general statements about the slowness of disks.
5	What is typically done to improve the performance of mechanical disks today in terms of access latencies and data transfers? State 2 approaches and explain them.	Hva gjøres vanligvis for å forbedre ytelsen til mekaniske disker i dag når det gjelder tilgangsforsinkelser og dataoverføringer? Nevn 2 tilnærmingen og forklar dem.	There are a lot of things that are attempted: (a) a large RAM or FLASH cache where tracks are kept when they have been accessed before, (b) data placement putting the more frequently read data onto those disk positions that are statistically closer to the disk head, (c) data prefetching into RAM, (d) making data in a file contiguous on disk to avoid a lot of arm movement, … ● 3: 2 approaches with explanations ● 2: 2 approaches with somewhat correct explanations

			● 1: 1 approach with somewhat correct explanation
6	Let us assume that we have a (very small) disk which has 50 tracks, numbered from 0 to 49, and which the disk head moves above and reads data from. Assume that at a given time, the disk head is positioned above and reading data from track 14. After this request, the following requests are in the disk scheduling queue (each number denotes the track on which the requested data block is stored, and the sequence shows the order in which the requests arrived in the queue, i.e., request 2 is the first that arrived in the system):	La oss anta at vi har en (veldig liten) disk som har 50 spor, nummerert fra 0 til 49, og som diskhodet beveger seg over og leser data fra. Anta at på et gitt tidspunkt er diskhodet plassert over og leser data fra spor 14. Etter denne forespørselen er følgende forespørsler i diskscheduleringskøen (hvert nummer angir sporet som den forespurte datablokken er lagret på, og sekvensen viser rekkefølgen forespørslene ankom i køen, dvs. forespørsel 2 er den første som ankom systemet):	 (last) 45 15 3 49 19 40 5 16 33 9 17 23 46 2 (first) 0 10 20 30 40 50  0 10 20 30 40 50 Assignment says: the head is currently moving from low-numbered tracks to high-numbered tracks Going upwards first, SCAN and LOOK are identical. Going in the wrong direction first, SCAN must go all the way to track 0.
	45 15 3 49 19 40 5 16 33 9 17 23 46 2		
	Make a graphical figure that shows how the disk head moves over the different tracks (in what order are request serviced) if we use “First-Come-First-Serve” (FCFS) and SCAN (the head is currently moving from low- numbered tracks to high-numbered tracks), respectively, and assume that all requests in the queue can be served during the same round.	Lag en grafisk figur som viser hvordan diskhodet beveger seg over de forskjellige sporene (i hvilken rekkefølge betjenes forespørselene) hvis vi bruker "First-Come- First-Serve" (FCFS) og SCAN (hodet beveger seg for øyeblikket fra lavt nummerert spor til høynummererte spor), henholdsvis, og forutsetter at alle forespørsler i køen kan betjenes i samme runde.	Drawing is the preferred response. Listing the steps in text form does not reduce the number of points. Minor mistakes (such as missing one or two numbers) are OK. 3: All correct. We accept going into the wrong direction first, but only if SCAN does correctly turn at track 0 and not at track 2. 2: Both attempted. Mostly correct but with some essential errors. Starting at one of the disk edges for SCAN gives at most one point. 1: The spirit of FCFS and SCAN is visible, but not much more; or only one of the parts is answered.

7-9 Routing

	Question in English	**Spørsmål på norsk**	**Retteveiledning**
7	 9 4 3 2 2 3 7 2 5 5 6 8 A B C D E F What is Dijkstra's Algorithm? Given the links in the figure with the distances between them, what is the best route from A to F based on Dijkstra's algorithm? In which order will the nodes B, C, D, E and F be visited?	Hva er Dijkstras algoritme? Gitt koblingene i bildet med avstander mellom dem, hva er den beste ruten fra A til F basert på Dijkstras algoritme? I hvilken rekkefølge vil nodene B, C, D, E og F besøkes.	Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph. The shortest path between A and F is through A-B-D-F with a total distance of 9. Starting from A, I will relabel all directly adjacent nodes with the distance to A. So D's label will be D(9,A), B's label will be B(3,A) and C's label will be C(5,A). I pick B and from B, I will relabel, C as C(6,B) and D as D(7,B), and I label E as E(10,B). I pick C and for D I choose the previous label as it is smaller, D (7,B), and I label E as E(10,C) and F as F(14,C). I pick D and relabel F as F(9,D) and E as E(9, D). This reaches the destination F. Note, while making the nodes label permanent, choose the smallest distance. Three components: (i) Disjkstra is explained, (ii) the problem is correctly solved (A-B-D-F) and the visited node order is correct (B-C-D-E-F). ● 3: Dijkstra's algorithm is explained, the problem is correctly solved and explained. ● 2: Two of the above are addressed.

			● 1: Only one of the above is addressed.
8	How does Distance Vector Routing work? List and explain the steps.	Hvordan fungerer en Distance Vector Routing? Oppgi og forklar stegene.	With Distance Vector Routing (DVR), every IS maintains a table (i.e., vector) stating the best known distance to all known destinations and the best line (direct neighbour) to be used on the way to the destination. Distance to direct neighours is measured. Distance to far-away neighbours is computed by adding measurements to neighbours' information. ISes then update tables by exchanging their complete routing information with their neighbors. DVR is imagined to work in rounds. ● 3: All DVR steps are described. ● 2: One of the main steps is missing or incorrect, for example that complete tables are sent, or that they are sent only to direct neighbours, or that the best route is determined by adding measurements to neighbours tables. ● 1: The DVR steps are buggy or incomplete, but DVR is still recognizable - especially in being different from broadcast-based Link State Routing.
9	You are the head of a company that is building a network in an underdeveloped area where resources such as energy are scarce. In this area, the power outages can also result in link failures. What are the 2 main properties you will consider for a	Du er leder for et selskap som bygger et nettverk i et underutviklet område hvor ressurser som energi er knappe. I dette området kan strømbruddene også føre til koblingsfeil. Hva er de 2 hovedegenskapene du vil vurdere for en rutingalgoritme som bør fungere godt innenfor dette området. Hvorfor?	● Robustness ○ Compensation for IS and link failures, especially due to outages ○ Handling of topology and traffic changes ● Simplicity ○ Minimize load of ISes since the resources are scarce

	routing algorithm that should work well within this area. Why?		● Optimality, especially in terms of overall energy usage ● Correctness/stability is important, even it is not specific to this problem ● 3: Have 2 of the above points correct with reasoning ● 2: Have 1 of the above points correct with reasoning ● 1: Have a few points listed without reasoning

10-12 Flow control

	Question in English	Spørsmål på norsk	Retteveiledning
Intro text	Flow control enables the sender to make sure that the receiver is not flooded. An important family of flow control mechanisms are the sliding window protocols. Sliding Window has further 3 different flavors: Go-Back-N, Selective Repeat and Credit Mechanism.	Flytkontroll gjør at senderen kan unngå å oversvømme mottakeren. En viktig familie av flytkontrollmekanismer er sliding window protokollene. Sliding window har ytterligere 3 forskjellige variasjoner: Go-Back-N, Selective Repeat og Credit Mechanism.	
10	How does Stop-and-wait work? What are the pros and cons of Stop-and-wait?	Hvordan funker Stop-and-wait? Hva er fordelene og ulemper med Stop-and-wait?	Stop-and-wait implements a timeout mechanism. Basic operation is as follows: 1. **Sender:** Transmits a single frame at a time. 2. Sender waits to receive ACK within time out. 3. **Receiver:** Transmits acknowledgement (ACK) when it receives a frame. 4. Go to **step 1** when ACK is received. 5. Repeat **step 1** (e.g. retransmit the frame) if a frame or ACK is lost during transmission and timeout is reached. Pros: very simple mechanism Cons: can be very inefficient as the sender needs to wait to receive an ACK for each successful transmission. This results in very long idle times and underutilization of the channel.

			3 points: Mechanism explained, pros/cons are discussed. 2 points: Mechanism explained but pros/cons are not discussed. 1 point: Mechanism not explained but pros/cons discussed.
11	Please explain how Go-back-N and Selective Repeat work. Which of these two sliding- window approaches would you use for wireless networks where random losses happens often (but usually only one packet loss at a time) and why?	Vennligst forklar hvordan Go-back-N or Selective Repeat fungerer. Hvilken av disse to sliding window-protokollene ville du brukt for trådløse nettverk da tilfeldig pakketap skjer ofte (men som oftest bare ett pakketap om gangen) og hvorfor?	Go-back-N: Receiver accepts packets only in order and discards all the packets arriving out of order. Therefore there is no buffer allocation. Selective Repeat: Receiver accepts the packets that arrive out of order, and buffers the packets but in a window. Therefore, there is a static buffer allocation. I would choose Selective Repeat as it uses bandwidth better when the network losses are random and only 1 packet in a row. 3 points: Mechanism explained, correct argument for the selection for random loss. 2 points: Mechanism explained but incorrect argument for the selection for random loss. 1 point: Mechanism not explained but correct argument for the selection for random loss.
12	What is a credit mechanism and how does it work? How do you handle situations where no credit remains?	Hva er en kredittmekanisme og hvordan fungerer den? Hvordan håndterer du situasjoner der ingen kreditt gjenstår?	The Credit mechanism is a sliding window based flow control where the receiver dynamically allocates the buffer based on the situation.

			The main principle: ● Sender requests required buffer amount ● Receiver reserves as many buffers as the actual situation permits ● Receiver returns ACKs and buffer-credits separately ○ ACK: confirmation only (does not imply buffer release) ○ CREDIT: buffer allocation ● Sender is blocked when all credits are used up 3 points: Credit mechanism is introduced and how it works is explained. Etter at kreditt har blitt brukt opp vil overføringen stoppe, når mottaker har ny bufferplass må den sende en ACK med gammel segmentnummer og ny kreditt. 2 points: Credit mechanism is not properly introduced but how it works is explained. 1 point: Credit mechanism is introduced but how it works is not explained.

13-15 Domain Name System

	Question in English	Spørsmål på norsk	Retteveiledning
Intro text	The Domain Name System (DNS) acts as an intermediary between the user and the internet, translating human-friendly domain names to machine-friendly IP addresses.	Domenenavnsystemet (DNS) fungerer som et mellomledd mellom brukeren og internett, og oversetter menneskevennlige domenenavn til maskinvennlige IP-adresser.	Each question addresses several elements of comprehension, indicated by a bracketed asterisk in the solutions below. Award one point for each element that has been fully and correctly explained. One half point may be awarded where the explanation is somewhat lacking in either fullness or correctness. Zero points should be given where an element is either wrongly explained or not explained at all. Negative points should not be given. The order in which elements are addressed does not matter. A maximum of three points may be awarded per question.
13	DNS maintains a hierarchical, as opposed to a flat, namespace. Explain the difference between these two ways of organizing a namespace and the motivation for this structure.	DNS har et hierarkisk, i motsetning til et flatt, navnerom. Forklar forskjellen mellom disse to måtene å organisere et navnerom på og motivasjonen for denne strukturen.	A flat namespace consists of a single level only. All names must be unique within the entire namespace. A hierarchical namespace consists of two or more levels DNS calls them domains. A name needs only be unique

			within its domain. One good motivation is distributed management (ownership), leading to better scaling. ● 3: difference is explained and at least one advantage (such as administration) is given ● 2: either the difference of organization or the motivation is given and correct ● 1: Student seems to have understood that DNS is hierarchical.
14	DNS constitutes a typically wide and shallow tree: nodes can have a very large number of children, but most leaf nodes are located not too far from the root (such as .com). Why does DNS exhibit this structure?	DNS utgjør et typisk bredt og grunt tre: noder kan ha et svært stort antall barn, men de fleste bladnoder er plassert ikke så langt fra roten (som f.eks. .com). Hvorfor fremviser DNS denne strukturen?	A fully qualified domain name should be short and memorable, that is, human friendly. Arbitrary words are allowed at every level (the vocabulary is huge), allowing very wide trees. 3: Student explains in some way that domain names with few elements are easier to remember and words can be long. 2: Allow for some confusion. Maybe the student talks about the hierarchical ownership of DNS. Maybe the student talks about the physical location of DNS servers. The explanation

			makes sense. 1: The same as 2, but the explanation makes no sense.
15	DNS queries can be handled in one of two ways: recursively or iteratively. Explain the difference between them, highlighting their pros and cons.	DNS-spørringer kan håndteres på én av to måter: rekursivt eller iterativt. Forklar forskjellen mellom dem, og fremhev deres fordeler og ulemper.	A recursive DNS query is sent from a local DNS server to a DNS root server (assuming no cache), which forwards the query down the DNS hierarchy, eventually reaching the DNS server responsible for the remote domain, whence the answer is returned along the same path back to the local DNS server. An iterative DNS query is sent from a local DNS server to a DNS root server (assuming no cache), which provides the address to a second-level DNS server so that the local DNS server can redirect its query. Redirection occurs at each level of the DNS hierarchy, eventually reaching the DNS server responsible for the remote domain, whence the answer is returned directy to the local DNS server. 3: The explanation is as above. 2: The explanation has mistakes. For example, caching only on the client, or asking the actual

			machine for its IP address (the low-level DNS servers should do this) 1: The answer has a sensible core but a lot of confusion. E.g., if the local DNS server is always asking root servers directly, or when root servers know every IP address in the world

16-18 Congestion control

	Question in English	**Spørsmål på norsk**	**Retteveiledning**
Intro text	Congestion control is essential to maintain proper operation of a computer network that does not have allocation of resources.	Metningskontroll er essensielt for å opprettholde normal operasjon av et datanettverk som ikke har allokering av ressurser.	
16	Briefly explain what congestion is and how it is dealt with by intermediate- and end- systems in the Internet.	Forklar kort what metningskontroll er og hvordan intermediate- og ende-systemer håndtere det i Internettet.	- Caused by input rate > output rate. - Difference between persistent and transient congestion. - Intermediate buffers - Intermediate drops/marks - End-system adjust sending rate 3 points: 4 / 5 discussed 2 points: 2 / 5 discussed 1 point: 1/ 5 discussed
17	Briefly explain how TCP New Reno congestion control is designed and which phases it consists of.	Forklar kort hvordan TCP New Reno metningskontroll er designet og hvilke faser den består av.	- Keep the number of in flight packets under a congestion window. - Slow start. IW increase by 1 for each ack, opt double every RTT, opt exponential growth. - Congestion avoidance. Increase by 1/W for each ack, opt increase by 1 every RTT, opt additive increase. - Congestion avoidance. Packet loss causes a multiplicative decrease and possibly a new slow start.

			- (Bonus) Distinguish between dup-ack and timeout. 3 points: all 4 bullet points are touched upon and are not clearly wrong. 2 points: Missing 2 of the 4 bullet-points 1 point: Getting 1 of the bullet-points right
18	Packet loss is used as a signal for TCP New Reno congestion control. Briefly explain how packet loss is detected and what weaknesses these detections have.	Pakketap blir brukt som et signal for TCP New Reno metningskontrol. Forklar kort hvordan pakketap blir detektert og hvilke svakheter disse deteksjonene har.	- Packet loss through timeout - Timeout needs an appropriate value - Packet loss through dup-acks - Reordering can case spurious signals (and retrans) 3 points: 3 / 4 discussed 2 points: 2 / 4 discussed 1 point: 1 / 4 discussed

19-21 Scheduling

	Question in English	Spørsmål på norsk	Retteveiledning
Intro text	For different kinds of computers and expected workload, we will use scheduling algorithms with very different properties. Please consider the following questions.	For ulike typer datamaskiner og forventet arbeidsmengde vil vi bruke scheduleringsalgoritmer med svært forskjellige egenskaper. Vennligst vurder følgende spørsmål.	
19	Describe how a Shortest-job-first (SJF) scheduler works. What is the advantage of SJF scheduling?	Beskriv hvordan en Shortest-job-first (SJF) scheduler fungerer. Hva er fordelen med SJF-schedulering?	For every new job, SJF determines how long it will run, and keep the ready queue sorted by runtime. SJF is simple and minimizes the average finishing time for jobs. ● 3: explanation as above ● 2: just the definition without stating the advantages of SJF ● 1: some confusion, for example if the student does not understand that SJF scheduling is not preemptive, they are not scheduled out and back in
20	Why is SJF scheduling not used as the main scheduling strategy for desktop computers that are used interactively for typical office workloads? Provide and explain one reason.	Hvorfor brukes ikke SJF-schedulering som den primære scheduleringsstrategien for stasjonære datamaskiner som brukes interaktivt for typisk kontorarbeid? Gi og forklar en grunn.	SJF makes sense only when the duration of jobs is known. In a normal desktop computer, we do not know that for the majority of jobs. Add on (only briefly mentioned in lecture): even if job time would be known, the need for multi-tasking would make it more sensible anyway to use SRTF (Shortest remaining time first) ● 3: It is sufficient to state that desktop

			computers need long-running tasks/processes with IO and not jobs without IO. There may be more or other reasons. ● 2: e.g.: The student does not understand that SJF cannot work with interactive workloads; the student may claim that very long processes/jobs can be starved. ● 1: at least one sensible fact about using SJF for tasks that must block because of IO
21	You manage a server that is meant for large, computation-heavy workloads without interactivity. Explain the kind of scheduling algorithm that you use on this server and why.	Du administrerer en server som er ment for stor, beregningstung arbeidslast uten interaktivitet. Forklar hva slags scheduleringsalgoritme du bruker på denne serveren og hvorfor.	The simplest answer is to use batch processing (ie. FIFO) because it has the small possible overhead and the least time to completion. One could also argue for SJF. Another answer could be the use of large time slices because overhead can be very small and time slices provide fairness. ● 3: a sensible answer as above ● 2: a misguided but coherent answer. E.g. a student has misunderstood “computation-heavy without interactivity”, but explains the properties of the workload - the scheduling algorithm makes sense for the student’s own scenario ● 1: the choice of scheduling algorithm is correct, but the explanation is missing or wrong. Example 3 poeng: FIFO + very few context switches, which leads to very little overhead. Example 1 poeng: FIFO + FIFO is easy to

			implement / FIFO has little runtime overhead.

22-24 Processes

	Question in English	**Spørsmål på norsk**	**Retteveiledning**
22	Explain what happens when a context switch between two processes happens because the timeslice of the running process is used up.	Forklar hva som skjer når en kontekstbytte mellom to prosesser skjer fordi tiden (timeslice) til den kjørende prosessen er brukt opp.	1. Scheduler stops the process A. 2. Store process A's state (like registers, instruction pointer) on stack or in PCB. 3. Set the process to the ready state and into the ready queue. 4. Restore the process state for the first process in the ready queues. 5. Recover process state for that process. 6. Start the timer for the new timeslices. 7. Give control to that process. Points: ● 3: at least the details above are given, maybe even more ● 2: some elements from the list above are missing, either the fact that the scheduler does the stopping or starting of processes, or the details related to state saving are missing. ● 1: the student writes something sensible about putting the old process into the ready queue and get a new process from there Students may be confused with switching

			because the process blocks on IO. That should remove one point, give 2 or 1 depending on the remaining details.
23	What can motivate a programmer to use multiple threads (kernel threads) instead of multiple processes?	Hva kan motivere en programmerer til å bruke flere tråder (kjernetråder) i steden for flere prosesser?	Only one reason is expected, but giving a lot of sensible motivations with limited explanation can compensate for a lack of depth. Example motivations: ● Processes have their own heap memory, threads do not. This makes communication difficult. ● Processes can not touch each other's regular memory, threads can (including each other's stack memory). Assuming a single reason: ● 3: an example is given, and the thread advantage is intuitive for the reader. ● 2: an example is given, but the explanation of the advantage is difficult to understand ● 1: an example is given without explanation
24	Assume that the computer has only a single processing core. When a new process is started using fork(), should the parent process continue to run, or should the newly created child process run first? Provide arguments for your decision.	Anta at datamaskinen bare har en enkel prosesseringskjerne. Når en ny prosess startes ved å bruke fork(), bør foreldreprosessen fortsette å kjøre, eller skal den nyopprettede barneprosessen kjøre først? Gi argumenter for avgjørelsen din.	There is no clear answer, but an OS must make a decision on this point. There are lots of possible answers. fork() is a system call, so there is no additional overhead either way. Child-after-fork makes sense when those jobs are usually small and cheap, for example in shell scripts (bash, perl). The child processes can often be so short that they create their output within one timeslice.

			Parent-after-fork makes sense when processes wait very soon after starting anyway. For example for servers that wait for new connections and dispatch children after accepting. They can quickly go back to select and context-switch there. ● 3 (a): the student does not make a choice and explains that both is advantageous in some situations ● 3 (b): the student makes a choice and has one convincing example why this choice is right. Remember that we assume that the scheduler in the kernel will always run when fork() is called, so the advantage must come later! A student might allow user-space waiting. ● 2: the student makes a choice but the explanation is not easily understandable or not convincing. ● 1: The student has understood what fork() does and that it is somehow related to waiting queues (ready queue) in the kernel, but not more.