IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE CS · 2023

IAE Academy free study material · Updated 27 October 2026

GATE CS 2023 Solved Paper

INSIDE

- All 65 questions, in the order of the paper
- The official answer to every question
- Step-by-step worked solutions
- The answer key on one page

Question paper and official answer key: IIT Kanpur

Online, with the questions linked to their topic notes: **iaeacademy.in/study/papers/cs-2023**

FREE COUNSELLING

Preparing for GATE CS? Talk to IAE Academy.

Or call **+91 84204 36524**

Every day, 10:30 am to 7:30 pm

Chat on WhatsApp

Free. Not for sale. Pass it on.

About this paper

65 questions, 100 marks. Question types: 34 MCQ · 15 MSQ · 16 NAT.

Official answer key: IIT Kanpur · https://gate.iitk.ac.in/GATE2023/doc/papers/2023/ANS/CS_ANS_GATE2023.pdf

We placed each question under a subject and topic ourselves; the exam paper does not label them.

This paper online, with the questions linked to their topic notes:
iaeacademy.in/study/papers/cs-2023

Answer key

Q	TYPE	MARKS	ANSWER
Q.1	MCQ	1	**B**
Q.2	MCQ	1	**Marks to all**
Q.3	MCQ	1	**A**
Q.4	MCQ	1	**A**
Q.5	MCQ	1	**C**
Q.6	MCQ	2	**D**
Q.7	MCQ	2	**C**
Q.8	MCQ	2	**D**
Q.9	MCQ	2	**B**
Q.10	MCQ	2	**A**
Q.11	MCQ	1	**B**
Q.12	MCQ	1	**B**
Q.13	MCQ	1	**D**
Q.14	MCQ	1	**C**
Q.15	MCQ	1	**A**
Q.16	MCQ	1	**A**
Q.17	MCQ	1	**B**
Q.18	MCQ	1	**B**
Q.19	MCQ	1	**C**
Q.20	MCQ	1	**C**
Q.21	MCQ	1	**B**
Q.22	MSQ	1	**B, C, D**
Q.23	MSQ	1	**C, D**
Q.24	MSQ	1	**A, C, D**
Q.25	MSQ	1	**A, C**
Q.26	MSQ	1	**B, C**
Q.27	MSQ	1	**A, C, D or C, D**
Q.28	MSQ	1	**B, D**
Q.29	MSQ	1	**A, C**
Q.30	NAT	1	**2, or 4**
Q.31	NAT	1	**0**
Q.32	NAT	1	**110**
Q.33	NAT	1	**2040**
Q.34	NAT	1	**10.2**
Q.35	NAT	1	**7**
Q.36	MCQ	2	**A**
Q.37	MCQ	2	**D**
Q.38	MCQ	2	**C**
Q.39	MCQ	2	**B**
Q.40	MCQ	2	**A**
Q.41	MCQ	2	**B**
Q.42	MCQ	2	**C**
Q.43	MCQ	2	**A**
Q.44	MCQ	2	**C**
Q.45	MCQ	2	**C**
Q.46	MCQ	2	**B**
Q.47	MCQ	2	**C**
Q.48	MCQ	2	**D**
Q.49	MSQ	2	**B, C, D**
Q.50	MSQ	2	**B, C, D**
Q.51	MSQ	2	**A, D**
Q.52	MSQ	2	**C, D**
Q.53	MSQ	2	**C, D**
Q.54	MSQ	2	**A, D**
Q.55	MSQ	2	**A, B**
Q.56	NAT	2	**5040**
Q.57	NAT	2	**4096**
Q.58	NAT	2	**5**
Q.59	NAT	2	**8**
Q.60	NAT	2	**2.374 to 2.376**
Q.61	NAT	2	**2**
Q.62	NAT	2	**6**
Q.63	NAT	2	**4**
Q.64	NAT	2	**19**
Q.65	NAT	2	**3**

General Aptitude Q.1 to Q.10

Q.1 MCQ · 1 MARK

General Aptitude › Grammar & vocabulary

We reached the station late, and _____ missed the train.

(A) near

(B) nearly

(C) utterly

(D) mostly

Official answer: **B**

SOLUTION

1. The blank comes just before the verb "missed", so it needs an adverb that modifies that verb.
2. (A) "near" is an adjective or preposition (or an adverb of place or time), not an adverb that can sit before "missed" here, so "near missed the train" is ungrammatical in this sentence.
3. (C) "utterly" means completely. "Utterly missed the train" is an odd collocation. It also does not fit the sense of reaching the station late, which suggests a narrow escape and not a total miss.
4. (D) "mostly" means for the most part. A train is either caught or missed, so "mostly missed" does not make sense.
5. (B) "nearly" means almost. "Nearly missed the train" is the natural phrase. Reaching the station late, we almost missed it.
6. So the answer is (B) nearly.

KEY IDEA
Choose the adverb that matches the sense of the sentence. Arriving late and then "nearly" missing the train means almost missing it, which is a narrow escape.

WATCH OUT FOR
Picking "utterly" or "mostly" because they are adverbs of degree. Neither fits a yes-or-no event such as missing a train, and "utterly" contradicts the idea of a narrow escape.

Q.2 MCQ · 1 MARK

General Aptitude › Reasoning

Kind : _____ : : Often : Frequently
(By word meaning)

(A) Mean

(B) Type

(C) Cruel

(D) Kindly

Official answer: **Marks to all**

Q.3 MCQ · 1 MARK

General Aptitude › Numerical

A series of natural numbers $F_1, F_2, F_3, F_4, F_5, F_6, F_7, \ldots$ obeys $F_{n+1} = F_n + F_{n-1}$ for all integers $n \geq 2$.

If $F_6 = 37$, and $F_7 = 60$, then what is F_1 ?

(A) 4

(B) 5

(C) 8

(D) 9

Official answer: **A**

SOLUTION

1. The rule $F_{n+1} = F_n + F_{n-1}$ can be run backwards: $F_{n-1} = F_{n+1} - F_n$.
2. With $n = 6$: $F_7 = F_6 + F_5$, so $F_5 = 60 - 37 = 23$.
3. With $n = 5$: $F_6 = F_5 + F_4$, so $F_4 = 37 - 23 = 14$.
4. With $n = 4$: $F_5 = F_4 + F_3$, so $F_3 = 23 - 14 = 9$.
5. With $n = 3$: $F_4 = F_3 + F_2$, so $F_2 = 14 - 9 = 5$.
6. With $n = 2$: $F_3 = F_2 + F_1$, so $F_1 = 9 - 5 = 4$.
7. Check going forward: $4, 5, 9, 14, 23, 37, 60$. Each term is the sum of the two before it, and all are natural numbers. So $F_1 = 4$, option (A).

KEY IDEA

A Fibonacci-type recurrence can be run backwards: $F_{n-1} = F_{n+1} - F_n$. Step back from the two known terms until you reach F_1.

WATCH OUT FOR

Subtracting in the wrong order, or stopping one term early. $F_2 = 5$ and $F_3 = 9$ are among the options, so stopping early gives a wrong answer.

Q.4 MCQ · 1 MARK

General Aptitude › Numerical

A survey for a certain year found that 90% of pregnant women received medical care at least once before giving birth. Of these women, 60% received medical care from doctors, while 40% received medical care from other healthcare providers.

Given this information, which one of the following statements can be inferred with certainty?

(A) More than half of the pregnant women received medical care at least once from a doctor.

(B) Less than half of the pregnant women received medical care at least once from a doctor.

(C) More than half of the pregnant women received medical care at most once from a doctor.

(D) Less than half of the pregnant women received medical care at most once from a doctor.

Official answer: **A**

SOLUTION

1. Let the total number of pregnant women be 100. Only percentages are given, so any base works. Then 90% of $100 = 90$ women received medical care at least once.
2. Of these 90 women, 60% received care from doctors: $0.6 \times 90 = 54$ women.
3. The other 40% (36 women) received care from other healthcare providers. Since $60\% + 40\% = 100\%$, the two groups together account for all 90 women who received care.
4. So 54 out of 100 pregnant women, that is 54%, received care from a doctor at least once. This is more than half, so (A) is certain.
5. (B) says less than half received care from a doctor. This is false, because $54\% > 50\%$.
6. (C) and (D) concern care received at most once from a doctor. The data give no information about how many times a woman saw a doctor, so neither can be inferred with certainty.

KEY IDEA
Percentages of a percentage multiply: 60% of the 90% who received care is 0.6 x 90% = 54% of all pregnant women. Compare that with 50%, and only accept statements that follow from the given data alone.

WATCH OUT FOR
Taking 60% of the pregnant women as the share who saw a doctor, and so forgetting that the 60% applies only to the 90% who received care. Another slip is reading "at most once" in (C) and (D) as if the data counted visits.

Q.5 MCQ · 1 MARK

General Aptitude › Spatial

Looking at the surface of a smooth 3-dimensional object from the outside, which one of the following options is TRUE?

(A) The surface of the object must be concave everywhere.

(B) The surface of the object must be convex everywhere.

(C) The surface of the object may be concave in some places and convex in other places.

(D) The object can have edges, but no corners.

Official answer: **C**

SOLUTION

1. Test each option with a counter-example: a statement that says "must" is false if even one smooth object breaks it.
2. (A) "Concave everywhere": seen from outside, a smooth ball bulges outward and is concave nowhere. This counter-example shows (A) is false.
3. (B) "Convex everywhere": a smooth doughnut (torus) has an inner ring where, seen from outside, the surface curves inward (it is saddle-shaped there). So the surface need not be convex everywhere, and (B) is false.
4. (C) "May be concave in some places and convex in other places": on a torus the outer part bulges out (convex) while the inner ring curves inward (concave), so one smooth object has both. The word "may" asks only for possibility, so (C) is true.
5. (D) "Edges, but no corners": an edge is a sharp crease, and a smooth surface has no sharp creases. So a smooth object cannot have edges, and (D) is false.
6. Only (C) holds.

KEY IDEA
A statement that says "must" is false if one counter-example exists. A statement that says "may" is true if one example exists. A smooth torus has both outward-bulging and inward-curving regions.

WATCH OUT FOR
Choosing (B) because a ball is convex, without testing shapes like a doughnut. Or choosing (D) without noticing that a smooth surface has no sharp edges at all.

Q.6 MCQ · 2 MARKS

General Aptitude › Reading

The country of Zombieland is in distress since more than 75% of its working population is suffering from serious health issues. Studies conducted by competent health experts concluded that a complete lack of physical exercise among its working population was one of the leading causes of their health issues. As one of the measures to address the problem, the Government of Zombieland has decided to provide monetary incentives to those who ride bicycles to work.

Based only on the information provided above, which one of the following statements can be logically inferred with certainty?

(A) All the working population of Zombieland will henceforth ride bicycles to work.

(B) Riding bicycles will ensure that all of the working population of Zombieland is free of health issues.

(C) The health experts suggested to the Government of Zombieland to declare riding bicycles as mandatory.

(D) The Government of Zombieland believes that riding bicycles is a form of physical exercise.

Official answer: **D**

SOLUTION

1. Rule for 'inferred with certainty': accept only what the passage states or forces. Reject anything that needs an extra assumption or goes beyond the passage.
2. (A) The government only offers incentives to those who ride bicycles to work. An incentive does not force everyone to ride, so 'all the working population will ride' is not certain.
3. (B) The passage says lack of exercise is one of the leading causes of the health issues, not the only cause. Riding bicycles therefore cannot be guaranteed to leave all of the working population free of health issues.
4. (C) Nothing in the passage says the experts suggested making cycling mandatory. The experts are mentioned only for their finding about the causes. The government's measure is an incentive, not a mandate.
5. (D) The government's measure is meant to address the problem, and the problem was traced to a complete lack of physical exercise. Offering incentives for riding bicycles as part of the remedy makes sense only if the government regards riding bicycles as a form of physical exercise. So this is the one statement the passage supports.
6. Hence the answer is (D).

KEY IDEA

For 'inferred with certainty' questions, pick the option that the passage's own logic forces. Reject options that predict outcomes for everyone, strengthen 'one of the causes' into 'the cause', or attribute claims to people the passage does not mention.

WATCH OUT FOR

Choosing (B) or (A) because they sound like the intended result of the policy. The passage says only that the incentive is offered and that lack of exercise is one of the causes, so neither a guaranteed cure nor universal cycling follows. Another slip is choosing (C) by assuming the experts recommended a mandate.

Q.7 MCQ · 2 MARKS

General Aptitude › Numerical

Consider two functions of time (t),

$$f(t) = 0.01\,t^2$$

$$g(t) = 4\,t$$

where $0 < t < \infty$.

Now consider the following two statements:

(i) For some $t > 0$, $g(t) > f(t)$.

(ii) There exists a T, such that $f(t) > g(t)$ for all $t > T$.

Which one of the following options is TRUE?

(A) only (i) is correct

(B) only (ii) is correct

(C) both (i) and (ii) are correct

(D) neither (i) nor (ii) is correct

Official answer: **C**

SOLUTION

1. Form the difference: $f(t) - g(t) = 0.01t^2 - 4t = 0.01\,t\,(t - 400)$.
2. For $t > 0$ the factor $0.01t$ is positive, so $f(t) - g(t)$ has the same sign as $t - 400$.
3. Statement (i): take $t = 1$. Then $g(1) = 4$ and $f(1) = 0.01$, so $g(1) > f(1)$. In fact $g(t) > f(t)$ for every $0 < t < 400$. So (i) is correct.
4. Statement (ii): for $t > 400$ we have $t - 400 > 0$, so $f(t) - g(t) > 0$, that is $f(t) > g(t)$.
5. Choose $T = 400$. Then $f(t) > g(t)$ for all $t > T$, so (ii) is correct.
6. Both statements hold, so the answer is option (C): both (i) and (ii) are correct.

KEY IDEA
A quadratic with a positive leading coefficient eventually overtakes any linear function. Here $f(t) - g(t) = 0.01t(t - 400)$ changes sign at $t = 400$, so $g > f$ before 400 and $f > g$ after it.

WATCH OUT FOR
Seeing the tiny coefficient 0.01 and concluding that f is always smaller than g. That is true only for small t, and the t^2 term wins once $t > 400$.

Q.8 MCQ · 2 MARKS

General Aptitude › Reading

Which one of the following sentence sequences creates a coherent narrative?
(i) Once on the terrace, on her way to her small room in the corner, she notices the man right away.
(ii) She begins to pant by the time she has climbed all the stairs.
(iii) Mina has bought vegetables and rice at the market, so her bags are heavy.
(iv) He was leaning against the parapet, watching the traffic below.

(A) (i), (ii), (iv), (iii)

(B) (ii), (iii), (i), (iv)

(C) (iv), (ii), (i), (iii)

(D) (iii), (ii), (i), (iv)

Official answer: **D**

SOLUTION

1. Find the opening sentence. Sentence (iii) is the only one that names the character (Mina) and sets the scene. Sentences (i), (ii) and (iv) use the pronouns she, she and he with no earlier noun to refer to, so none of them can open the passage.
2. Find what follows (iii). Mina's bags are heavy, which explains why she is out of breath in (ii): 'She begins to pant by the time she has climbed all the stairs'. Here 'she' refers to Mina.
3. Find what follows (ii). Having climbed the stairs, she arrives on the terrace. This matches (i): 'Once on the terrace … she notices the man right away'. 'Once on' signals arrival after the climb.
4. Place (iv) last. 'He' is the man she has just noticed in (i), and 'He' has no antecedent before (i), so (iv) must come after (i).
5. The order by time and pronoun reference is (iii), (ii), (i), (iv). This is option (D).
6. Eliminate the rest. (A) opens with (i), whose 'she' has no antecedent, and it puts (iii) last, although (iii) introduces Mina. (B) opens with (ii), whose 'she' is unexplained. (C) opens with (iv), whose 'He' has no antecedent.

KEY IDEA
In sentence-ordering questions, begin with the sentence that introduces the subject by name, because pronouns need an earlier noun to refer to. Then follow the cause-and-effect and time order of the events (bags, stairs, terrace, man).

WATCH OUT FOR
Opening with (i), because it sets a scene on the terrace, or with (iv). Both use pronouns ('she', 'He') that refer to someone not yet introduced. Another slip is to put (iv) before (i), so that 'He' appears before the man has been noticed.

Q.9 MCQ · 2 MARKS

General Aptitude › Reasoning

$f(x)$ and $g(y)$ are functions of x and y, respectively, and $f(x) = g(y)$ for all real values of x and y. Which one of the following options is necessarily TRUE for all x and y?

(A) $f(x) = 0$ and $g(y) = 0$

(B) $f(x) = g(y) =$ constant

(C) $f(x) \neq$ constant and $g(y) \neq$ constant

(D) $f(x) + g(y) = f(x) - g(y)$

Official answer: **B**

SOLUTION

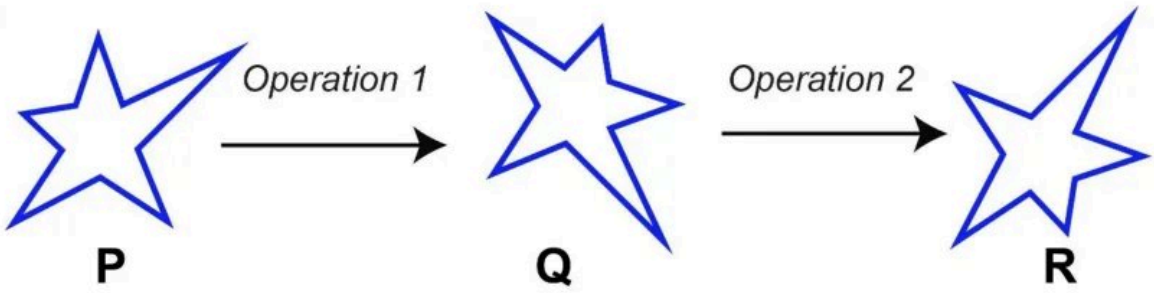
1. f depends only on x and g depends only on y, and $f(x) = g(y)$ must hold for every pair (x, y) of real numbers.
2. Fix $y = y_0$. Then $f(x) = g(y_0)$ for every x, and $g(y_0)$ does not change with x. So f takes one fixed value at all x: f is constant.
3. Fix $x = x_0$. Then $g(y) = f(x_0)$ for every y, so g is constant too.
4. Writing $f(x) = c_1$ and $g(y) = c_2$, the condition at any one pair gives $c_1 = c_2$. So $f(x) = g(y) = c$ for some constant c. This is option (B).
5. Option (A) is not necessarily true: c can be any number, such as 5, not only 0.
6. Option (C) is false: it says both functions are not constant, which contradicts what we derived.
7. Option (D) simplifies to $2g(y) = 0$, that is $g(y) = 0$. This holds only when $c = 0$, so it is not necessarily true.

KEY IDEA
If a function of x alone equals a function of y alone for all independent real x and y, neither can vary, so both equal the same constant.

WATCH OUT FOR
Choosing (A) or (D) by assuming the common constant must be zero. The constant can be any real number, so only "constant" is guaranteed.

Q.10 MCQ · 2 MARKS

General Aptitude › Spatial

Q.10	Which one of the options best describes the transformation of the 2-dimensional figure **P** to **Q**, and then to **R**, as shown?
	
(A)	*Operation 1*: A clockwise rotation by 90° about an axis perpendicular to the plane of the figure *Operation 2*: A reflection along a horizontal line
(B)	*Operation 1*: A counter clockwise rotation by 90° about an axis perpendicular to the plane of the figure *Operation 2*: A reflection along a horizontal line
(C)	*Operation 1*: A clockwise rotation by 90° about an axis perpendicular to the plane of the figure *Operation 2*: A reflection along a vertical line
(D)	*Operation 1*: A counter clockwise rotation by 180° about an axis perpendicular to the plane of the figure *Operation 2*: A reflection along a vertical line

Official answer: **A**

SOLUTION

1. Follow the one distinctive feature, the long spike. In P it points to the upper right, roughly east-north-east.
2. In Q the long spike points to the lower right, roughly south-south-east.
3. Going from east-north-east to south-south-east is a turn of about 90 degrees clockwise (east towards south). A counter clockwise turn of 90 degrees would send the spike to the north-north-west, and a

turn of 180 degrees would send it to the left. So Operation 1 is a clockwise rotation by 90 degrees about an axis perpendicular to the plane, which leaves options (A) and (C).

4. In R the long spike points to the upper right, while in Q it pointed to the lower right. The spike stays on the right and only its up/down sense is swapped.
5. A reflection along a horizontal line (a mirror across a horizontal axis) swaps top and bottom and keeps left and right. It takes a lower-right spike to an upper-right one, as in R.
6. A reflection along a vertical line would swap left and right instead, sending the lower-right spike to the lower left, which does not match R. So Operation 2 is a reflection along a horizontal line.
7. Operation 1 is a clockwise rotation by 90 degrees and Operation 2 is a reflection along a horizontal line. This is option (A).

KEY IDEA

Follow one distinctive feature (the long spike) through each step. A rotation changes its direction by a fixed angle. A reflection along a horizontal line swaps up and down only, and a reflection along a vertical line swaps left and right only.

WATCH OUT FOR

Mixing up the two reflections. A reflection along a horizontal line flips the figure top to bottom, not left to right. The other slip is reading the rotation direction the wrong way: from east-north-east to south-south-east is clockwise.

Computer Science and Information Technology Q.11 to Q.65

Q.11 MCQ · 1 MARK

Compiler Design › Lexical analysis

Consider the following statements regarding the front-end and back-end of a compiler.
S1: The front-end includes phases that are independent of the target hardware.
S2: The back-end includes phases that are specific to the target hardware.
S3: The back-end includes phases that are specific to the programming language used in the source code.
Identify the CORRECT option.

(A) Only S1 is TRUE.

(B) Only S1 and S2 are TRUE.

(C) S1, S2, and S3 are all TRUE.

(D) Only S1 and S3 are TRUE.

Official answer: **B**

SOLUTION

1. A compiler is split into a front-end (analysis) and a back-end (synthesis). The front-end covers lexical analysis, parsing, semantic analysis and intermediate-code generation.
2. The front-end works from the source language and produces an intermediate representation. It does not depend on the target machine, so S1 is TRUE.
3. The back-end covers code optimisation and target code generation, including instruction selection, register allocation and instruction scheduling. These depend on the target hardware, so S2 is TRUE.
4. The language-specific work (syntax, type rules, scoping) is done in the front-end. The back-end takes a language-independent intermediate representation, so S3 is FALSE.
5. Only S1 and S2 are true, which is option (B).

KEY IDEA
The front-end is tied to the source language and independent of the target machine. The back-end is tied to the target machine and independent of the source language.

WATCH OUT FOR
Thinking the back-end also holds language-specific phases, because it follows the front-end in the pipeline. The back-end only sees the intermediate representation, which has no source-language details.

Q.12 MCQ · 1 MARK

Data Structures and Algorithms › Trees & heaps

Which one of the following sequences when stored in an array at locations $A[1], \ldots, A[10]$ forms a max-heap?

(A) 23, 17, 10, 6, 13, 14, 1, 5, 7, 12

(B) 23, 17, 14, 7, 13, 10, 1, 5, 6, 12

(C) 23, 17, 14, 6, 13, 10, 1, 5, 7, 15

(D) 23, 14, 17, 1, 10, 13, 16, 12, 7, 5

Official answer: **B**

SOLUTION

1. A max-heap stored 1-indexed in $A[1..10]$ requires, for every node i that has children, $A[i] \geq A[2i]$ and $A[i] \geq A[2i+1]$ (node 5 has only child index 10; nodes 6–10 are leaves since $2 \times 6 = 12 > 10$).
2. Option A: node 3, $A[3] = 10$, has child $A[6] = 14$. Since $10 < 14$, the property fails: reject.
3. Option B: check all internal nodes. $A[1] = 23 \geq A[2] = 17$, $A[3] = 14$;
   $A[2] = 17 \geq A[4] = 7$, $A[5] = 13$; $A[3] = 14 \geq A[6] = 10$, $A[7] = 1$; $A[4] = 7 \geq A[8] = 5$, $A[9] = 6$;

$A[5] = 13 \geq A[10] = 12$. All pairs satisfy the property: B is a valid max-heap.

4. Option C: node 5, $A[5] = 13$, has child $A[10] = 15$. Since $13 < 15$, the property fails: reject.
5. Option D: node 4, $A[4] = 1$, has children $A[8] = 12, A[9] = 7$. Since $1 < 12$, the property fails: reject.
6. Only option B satisfies the max-heap property everywhere.

KEY IDEA

In a 1-indexed array max-heap, node i's children sit at $2i$ and $2i + 1$, and the heap property requires every parent's value to be greater than or equal to both of its children's values, everywhere in the array. Checking this level by level (or bottom-up from the last internal node) is enough to rule options in or out.

WATCH OUT FOR

Treating the array as 0-indexed (children at $2i + 1, 2i + 2$) instead of 1-indexed, which shifts every parent-child pair and can make a genuine violation look fine, or a valid heap look broken.

Q.13 MCQ · 1 MARK

Data Structures and Algorithms › Linear structures

Let `SLLdel` be a function that deletes a node in a singly-linked list given a pointer to the node and a pointer to the head of the list. Similarly, let `DLLdel` be another function that deletes a node in a doubly-linked list given a pointer to the node and a pointer to the head of the list.

Let n denote the number of nodes in each of the linked lists. Which one of the following choices is TRUE about the worst-case time complexity of `SLLdel` and `DLLdel`?

(A) SLLdel is $O(1)$ and DLLdel is $O(n)$

(B) Both SLLdel and DLLdel are $O(\log(n))$

(C) Both SLLdel and DLLdel are $O(1)$

(D) SLLdel is $O(n)$ and DLLdel is $O(1)$

Official answer: **D**

SOLUTION

1. Deleting a node requires updating its predecessor's link (in a DLL, also the successor's prev link) to bypass the deleted node. The actual pointer surgery is O(1) once the predecessor is known.
2. In a singly-linked list, each node stores only a `next` pointer; there is no way to reach a node's predecessor from the node itself.
3. Hence `SLLdel`, given only a pointer to the node and the head, must traverse from the head to locate the predecessor before it can unlink the node. In the worst case (the target node near the tail), this traversal visits Θ(n) nodes, so SLLdel is $O(n)$.
4. In a doubly-linked list, each node stores a `prev` pointer in addition to `next`, so the predecessor is available directly from the given node pointer, and no traversal is needed.
5. `DLLdel` can therefore immediately update `prev→next` and `next→prev` to unlink the node, doing constant work regardless of n, so DLLdel is $O(1)$.
6. Thus SLLdel is $O(n)$ and DLLdel is $O(1)$, which is option (D).

KEY IDEA

To unlink a node from a linked list you must update the predecessor's link field. A singly-linked node stores no back-pointer, so given only the node itself, finding its predecessor forces a scan from the head: O(n) worst case. A doubly-linked node already stores `prev`, so the predecessor is known instantly and unlinking (fixing `prev→next` and `next→prev`) is O(1).

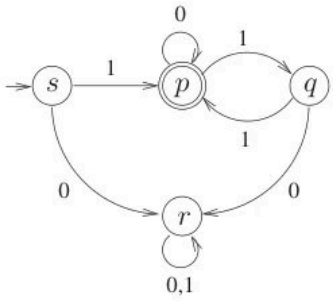
WATCH OUT FOR

Assuming that since we're "given a pointer to the node," deletion can always be done in O(1) by copying the next node's data into the given node and deleting the next node (a common trick for other problems), but here it must actually delete the node given (e.g., if it's the last node in an SLL, that trick fails, and worst case still requires finding the predecessor via head traversal since a plain singly-linked node has no back-link).

Q.14 MCQ · 1 MARK

Theory of Computation › Finite automata

Q.14 Consider the Deterministic Finite-state Automaton (DFA) $\mathcal{A}$ shown below. The DFA runs on the alphabet $\{0, 1\}$, and has the set of states $\{s, p, q, r\}$, with s being the start state and p being the only final state.



Which one of the following regular expressions correctly describes the language accepted by $\mathcal{A}$?

(A)	$1(0^*11)^*$
(B)	$0(0+1)^*$
(C)	$1(0+11)^*$
(D)	$1(110^*)^*$

Official answer: **C**

SOLUTION

1. Read the transitions from the figure: $\delta(s,1) = p$, $\delta(s,0) = r$, $\delta(p,0) = p$, $\delta(p,1) = q$, $\delta(q,1) = p$, $\delta(q,0) = r$, $\delta(r,0) = \delta(r,1) = r$. The start state is s and the only final state is p.
2. State r is a trap (dead) state: once the machine enters it, it never leaves, so no accepted string passes through r. From s, the symbol 0 leads to r, so every accepted string must begin with 1 and move to p.
3. Every accepted string ends in p. Looking at the loops through p that avoid r: reading 0 from p returns to p. Reading 1 from p goes to q. From q, reading 0 goes to r, so the only usable move is 1, back to p. The ways to go from p back to p are therefore the blocks 0 and 11.
4. The language is the first symbol 1 followed by any sequence of blocks 0 or 11:

$$L(\mathcal{A}) = 1(0+11)^*.$$

5. Check the options on short strings. (B) begins with 0, which the DFA sends to r and rejects, so (B) is wrong. (A) $1(0^*11)^*$ does not generate 10: after the leading 1 the remaining 0 is not of the form 0^*11. The DFA accepts 10 ($s \to p \to p$), so (A) is wrong. (D) $1(110^*)^*$ also does not generate 10, so (D) is wrong. (C) generates 1, 10, 111, 1011, 1110, and the DFA accepts each of these. It also rejects 11, 101 and 1101 (they end in q), and (C) cannot generate them either. So (C) is correct.

KEY IDEA

Remove the dead state r and look at the loops through the final state p. Each return to p is either the block 0 (self-loop) or the block 11 (through q). The string must start with 1.

WATCH OUT FOR

Writing the loop at p as 0^* attached to 11, or as 110^*. This forces every 0 to sit next to a 11 block and wrongly rejects strings such as 10. A 0 can be read at p on its own, independently of the 11 blocks.

Q.15 MCQ · 1 MARK

Discrete Mathematics › Recurrences

The Lucas sequence L_n is defined by the recurrence relation: $L_n = L_{n-1} + L_{n-2}$, for $n \geq 3$, with $L_1 = 1$ and $L_2 = 3$.

Which one of the options given is TRUE?

(A) $L_n = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n$

(B) $L_n = \left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{3}\right)^n$

(C) $L_n = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{3}\right)^n$

(D) $L_n = \left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n$

Official answer: **A**

SOLUTION

1. The characteristic equation of $L_n = L_{n-1} + L_{n-2}$ is $x^2 = x + 1$, that is $x^2 - x - 1 = 0$.
2. Its roots are $\varphi = \frac{1+\sqrt{5}}{2}$ and $\psi = \frac{1-\sqrt{5}}{2}$, with $\varphi + \psi = 1$ and $\varphi\psi = -1$. These are distinct, so the general solution is $L_n = a\varphi^n + b\psi^n$.
3. Options B and C contain $\frac{1-\sqrt{5}}{3}$, which is not a root of $x^2 - x - 1 = 0$. These expressions do not satisfy the recurrence, so B and C are false.
4. The initial values require $a\varphi + b\psi = 1$ and $a\varphi^2 + b\psi^2 = 3$. Because $\varphi \neq \psi$, this linear system has exactly one solution (a, b).
5. Try $a = b = 1$ (option A). First, $L_1 = \varphi + \psi = 1$, which matches.
6. Next, $L_2 = \varphi^2 + \psi^2 = (\varphi + \psi)^2 - 2\varphi\psi = 1 + 2 = 3$, which matches. Since the solution is unique, $a = b = 1$ and $L_n = \varphi^n + \psi^n$.
7. Check option D, which has $a = 1$, $b = -1$: $L_1 = \varphi - \psi = \sqrt{5} \neq 1$, so D fails.
8. As a further check, $L_3 = L_2 + L_1 = 4$, and $\varphi^3 + \psi^3 = (\varphi + \psi)^3 - 3\varphi\psi(\varphi + \psi) = 1 + 3 = 4$. The answer is option A.

KEY IDEA
Solve the linear recurrence through its characteristic roots φ, ψ, then fit the two initial values. Because the roots are distinct, the fit is unique, so a candidate that satisfies both initial values is the answer.

WATCH OUT FOR
Picking D because it looks like the familiar Fibonacci (Binet) formula. That has a minus sign and a $\sqrt{5}$ in the denominator. For Lucas numbers the two terms are simply added with coefficients 1 and 1. Options B and C can be dismissed at once because their second base is not a root.

Q.16 MCQ · 1 MARK

Database Management and Warehousing › Relational algebra

Which one of the options given below refers to the degree (or arity) of a relation in relational database systems?

(A) Number of attributes of its relation schema.

(B) Number of tuples stored in the relation.

(C) Number of entries in the relation.

(D) Number of distinct domains of its relation schema.

Official answer: **A**

SOLUTION

1. The **degree** (arity) of a relation is the number of columns, which is the number of attributes in its relation schema.
2. The number of tuples (rows) stored in a relation is its **cardinality**, not its degree. Option (B) describes cardinality.
3. "Number of entries" is not a standard term for a relation. If it means cells, the count is degree $\times$ cardinality, which is neither the degree nor the cardinality. Option (C) is wrong.
4. Distinct domains do not give the degree. Two attributes can share one domain, for example two date columns, so the number of distinct domains can be smaller than the number of attributes. Option (D) is wrong.
5. Hence the degree is the number of attributes of the relation schema, option (A).

KEY IDEA
Degree is the number of columns (attributes). Cardinality is the number of rows (tuples).

WATCH OUT FOR
Mixing up degree and cardinality, and picking the number of tuples (B) because it sounds like the size of the relation.

Q.17 MCQ · 1 MARK

Computer Networks › Performance

Suppose two hosts are connected by a point-to-point link and they are configured to use Stop-and-Wait protocol for reliable data transfer. Identify in which one of the following scenarios, the utilization of the link is the lowest.

(A) Longer link length and lower transmission rate

(B) Longer link length and higher transmission rate

(C) Shorter link length and lower transmission rate

(D) Shorter link length and higher transmission rate

Official answer: **B**

SOLUTION

1. Stop-and-Wait sends one frame and then waits for its acknowledgement, so the link carries data only during the transmission time T_t in each cycle of length $T_t + 2T_p$.
2. Utilization $U = \dfrac{T_t}{T_t + 2T_p} = \dfrac{1}{1 + 2a}$, where $a = \dfrac{T_p}{T_t}$.
3. Here $T_t = \dfrac{L}{B}$ (frame size over transmission rate) and $T_p = \dfrac{d}{v}$ (link length over propagation speed), so $a = \dfrac{d\,B}{v\,L}$.
4. U falls as a rises. a grows with a longer link (d larger) and with a higher transmission rate (B larger), since the frame size L and speed v do not change between the options.
5. So U is lowest when the link is longer and the transmission rate is higher, which is option (B).
6. Check the other options. In (A) and (D) one factor raises a and the other lowers it, so a is in between. In (C) both factors lower a, so (C) has the highest utilization.

KEY IDEA

In Stop-and-Wait, $U = \dfrac{1}{1 + 2a}$ with $a = T_p/T_t$. Utilization is lowest when a is largest, that is, a long propagation delay and a short transmission time (a fast link).

WATCH OUT FOR

Thinking a faster link always improves utilization. A higher rate shortens T_t but leaves T_p unchanged, so the sender idles for a larger share of each cycle and U drops.

Q.18 MCQ · 1 MARK

Engineering Mathematics › Linear algebra

Let

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{bmatrix}.$$

Let det(A) and det(B) denote the determinants of the matrices A and B, respectively. Which one of the options given below is TRUE?

(A) det(A) = det(B)

(B) det(B) = -det(A)

(C) det(A) = 0

(D) det(AB) = det(A) + det(B)

Official answer: **B**

SOLUTION

1. Write the rows of A as $R_1 = (1, 2, 3, 4)$, $R_2 = (4, 1, 2, 3)$, $R_3 = (3, 4, 1, 2)$, $R_4 = (2, 3, 4, 1)$.

2. The rows of B, from top to bottom, are $(3,4,1,2) = R_3$, $(4,1,2,3) = R_2$, $(1,2,3,4) = R_1$, $(2,3,4,1) = R_4$. So B is A with rows 1 and 3 interchanged.
3. Interchanging two rows multiplies the determinant by -1, so $\det(B) = -\det(A)$. Option (B) is true.
4. Check the other options. A is circulant: each row is the previous row shifted right by one place. So $A = I + 2P + 3P^2 + 4P^3$ with P the cyclic shift matrix, and its eigenvalues are $\lambda_k = 1 + 2\omega^k + 3\omega^{2k} + 4\omega^{3k}$ with $\omega = i$.
5. $\lambda_0 = 10$, $\lambda_2 = 1 - 2 + 3 - 4 = -2$, $\lambda_1 = 1 + 2i - 3 - 4i = -2 - 2i$, $\lambda_3 = 1 - 2i - 3 + 4i = -2 + 2i$.
6. $\det(A) = 10 \cdot (-2) \cdot \left((-2)^2 + 2^2\right) = -160$. So $\det(A) \neq 0$ and (C) is false.
7. Then $\det(B) = 160$. Since $-160 \neq 160$, (A) is false.
8. For (D): $\det(AB) = \det(A)\det(B) = -25600$, while $\det(A) + \det(B) = 0$. So (D) is false.
9. Only (B) is true.

KEY IDEA

B is A with two rows interchanged, and a row swap flips the sign of the determinant. No determinant has to be computed to pick the true option.

WATCH OUT FOR

Expanding both 4x4 determinants by hand and making an arithmetic slip, or taking det(A) = det(B) because the two matrices contain the same entries.

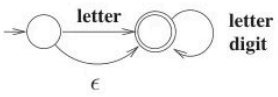
Q.19 MCQ · 1 MARK

Q.19 Consider the following definition of a lexical token **id** for an identifier in a programming language, using extended regular expressions:

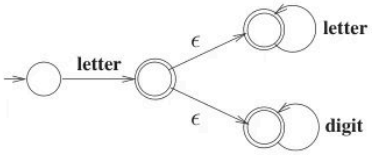
$$\begin{aligned} \textbf{letter} &\rightarrow \texttt{[A-Za-z]} \\ \textbf{digit} &\rightarrow \texttt{[0-9]} \\ \textbf{id} &\rightarrow \textbf{letter}\ (\textbf{letter} \mid \textbf{digit})^* \end{aligned}$$

Which one of the following Non-deterministic Finite-state Automata with ϵ-transitions accepts the set of valid identifiers? (A double-circle denotes a final state)

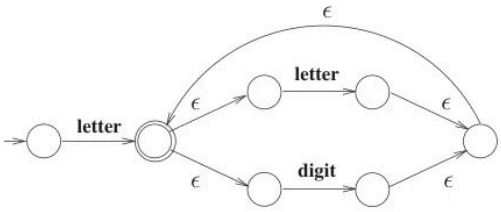
(A)



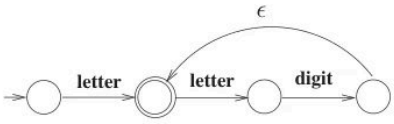
(B)



(C)



(D)



Official answer: **C**

SOLUTION

1. The target language is $L = \text{letter}(\text{letter} \mid \text{digit})^*$: one letter first, then any mix of letters and digits, in any order, of any length (including none).
2. (A): the start state q_0 has an ε-move to the final state q_1, so the empty string ε is accepted. Since $\varepsilon \notin L$, (A) is wrong.

3. (B): after the first letter, the machine takes an ε-move to one of two separate final states. One loops only on letter, the other only on digit, so it accepts $\text{letter}\ \text{letter}^*$ or $\text{letter}\ \text{digit}^*$ but never a mix. The string letter digit letter is rejected although it is in L, so (B) is wrong.
4. (D): after the first letter the machine reaches the final state q_1, and from there it can only read letter then digit before an ε-move returns it to q_1. It accepts $\text{letter}(\text{letter}\ \text{digit})^*$. The string letter digit, which is in L, is rejected, so (D) is wrong.
5. (C): the first letter leads to the final state q_1. From q_1 two ε-branches start, one reading letter and one reading digit. Both end in a non-final state that has an ε-move back to q_1.
6. So each trip around the loop through q_1 reads exactly one letter or one digit, in any order and any number of times, and q_1 is final. This is the standard construction for $(\text{letter} \mid \text{digit})^*$ placed after the first letter, so the machine accepts exactly L.
7. Hence the answer is (C).

KEY IDEA

A letter, then a loop of choices (letter | digit) that returns to a final state, is how $\text{letter}(\text{letter} \mid \text{digit})^*$ is built. Check each option for ε, for a mixed string like letter digit letter, and for a single letter followed by a digit.

WATCH OUT FOR

Choosing (B) because it has both a letter loop and a digit loop. The two loops sit on separate final states, so a letter and a digit can never be mixed after the first letter.

Q.20 MCQ · 1 MARK

Data Structures and Algorithms › Sorting & hashing

An algorithm has to store several keys generated by an adversary in a hash table. The adversary is malicious who tries to maximize the number of collisions. Let k be the number of keys, m be the number of slots in the hash table, and $k > m$.

Which one of the following is the best hashing strategy to counteract the adversary?

(A) Division method, i.e., use the hash function $h(k) = k \mod m$.

(B) Multiplication method, i.e., use the hash function $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$, where A is a carefully chosen constant.

(C) Universal hashing method.

(D) If k is a prime number, use Division method. Otherwise, use Multiplication method.

Official answer: **C**

SOLUTION

1. The division method h(k) = k mod m and the multiplication method h(k) = ⌊m(kA − ⌊kA⌋)⌋ are both single, fixed functions once m (and A) are chosen, and this choice is known in advance to the adversary.
2. Since k > m keys must be placed into m slots under a known, fixed function, the pigeonhole principle guarantees the adversary can choose k keys that all map to the same slot (e.g. all keys congruent mod m), forcing Θ(k) collisions in the worst case. The same vulnerability applies to the multiplication method once A is fixed.
3. Option D only switches between these two still-fixed, still-predictable functions based on whether k is prime (a public, adversary-knowable rule), so it inherits the same weakness and offers no real protection.
4. Universal hashing instead selects the hash function uniformly at random, at run time, from a carefully designed family of functions: a choice made independently of, and unknown to, the adversary, who must fix the keys before the function is drawn.

5. Because the adversary cannot know which function will be used when choosing the keys, no fixed set of k keys can be guaranteed in advance to collide under the eventually-drawn function.
6. The standard guarantee for a universal family is that, for any fixed set of n distinct keys, the expected number of pairwise collisions is O(n/m): a probabilistic guarantee that holds regardless of how the adversary chose the keys.
7. Hence Universal hashing (option C) is the hashing strategy that provably counteracts an adversary who chooses keys to maximize collisions.

KEY IDEA

Any fixed (deterministic) hash function can be defeated by an adversary who knows it in advance: since $k > m$, by pigeonhole the adversary can always choose k keys that all map to the same slot under a known fixed function, forcing worst-case $\Theta(k)$ collisions. Universal hashing defeats this by choosing the hash function *randomly at run time* from a well-designed family, independent of and unknown to the adversary when the keys are chosen, giving a provable expected-collision guarantee no matter what keys are picked.

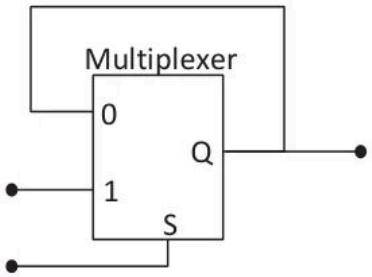
WATCH OUT FOR

Assuming that any single, fixed hash function (division method, multiplication method, or a rule like option D that just switches between two fixed functions based on a property of k) is safe merely because it "spreads keys well on average": an adversary who knows the fixed function in advance can always construct k > m keys that collide, regardless of how good the function looks statistically.

Q.21 MCQ · 1 MARK

Digital Electronics › Sequential

Q.21 The output of a 2-input multiplexer is connected back to one of its inputs as shown in the figure.



Match the functional equivalence of this circuit to one of the following options.

(A) D Flip-flop

(B) D Latch

(C) Half-adder

(D) Demultiplexer

Official answer: **B**

SOLUTION

1. Read the wiring from the figure: data input 1 is an external input, call it D. The select line S is external. Data input 0 is fed from the output Q, so the output feeds back into the multiplexer.
2. A 2-to-1 multiplexer gives $Q = \overline{S} \cdot I_0 + S \cdot I_1$. With $I_0 = Q$ (feedback) and $I_1 = D$ this becomes
$$Q = \overline{S}\, Q + S\, D.$$
3. Case $S = 1$: $Q = D$. The output follows the input, so the circuit is transparent.
4. Case $S = 0$: $Q = Q$. The output keeps its previous value, so the circuit holds (stores) its state.
5. The behaviour is transparent while S is high and holding while S is low. That is a level-sensitive D latch with S as the enable and D as the data input, so the answer is (B).
6. Why not the others: a D flip-flop updates only at a clock edge, not for the whole time a level is high. A half-adder has two outputs (sum and carry) and no memory. A demultiplexer has one data input and several outputs and has no feedback path.

KEY IDEA
Write the multiplexer equation with the output fed back into one input: $Q = \overline{S}Q + SD$. It follows D when $S = 1$ and holds when $S = 0$, which is a level-triggered D latch with S as the enable.

WATCH OUT FOR
Calling it a D flip-flop because it stores a bit. A flip-flop samples only on a clock edge. This circuit is transparent for the whole time $S = 1$, so it is a latch.

Q.22 MSQ · 1 MARK

Operating Systems › Processes & threads

Which one or more of the following need to be saved on a context switch from one thread (T1) of a process to another thread (T2) of the same process?

(A) Page table base register

(B) Stack pointer

(C) Program counter

(D) General purpose registers

Official answer: **B, C, D**

SOLUTION

1. Threads of one process share one address space, and so share one page table. Only each thread's own execution state has to be saved on a thread-to-thread switch.
2. Each thread has its own execution state: the program counter (where T1 resumes), the stack pointer (each thread has its own stack), and the general purpose registers (T1's live values). All three are saved when T1 is switched out.
3. The page table base register points to the page table of the address space. T1 and T2 belong to the same process, so this value is the same for both and does not change on the switch. It need not be saved. This is the difference from a process switch, where the address space changes.
4. So the items to save are the stack pointer, the program counter and the general purpose registers: options B, C and D.

KEY IDEA
A thread switch within one process saves only per-thread state: program counter, stack pointer and general purpose registers. The address space, and so the page table base register, is shared and unchanged.

WATCH OUT FOR
Treating a thread switch like a process switch and choosing the page table base register. That register changes only when the address space changes, which does not happen between threads of the same process.

Q.23 MSQ · 1 MARK

Operating Systems › Processes & threads

Which one or more of the following options guarantee that a computer system will transition from user mode to kernel mode?

(A) Function Call

(B) `malloc` Call

(C) Page Fault

(D) System Call

Official answer: **C, D**

SOLUTION

1. A switch from user mode to kernel mode happens only through a trap (system call), an interrupt or an exception (such as a page fault). The question asks which events are certain to cause one.

2. (A) An ordinary function call is a jump within user code. It stays in user mode, so it does not guarantee a switch.
3. (B) `malloc` is a C library routine. It usually serves requests from the heap the library already holds. It calls `brk` or `mmap` only when the heap must grow, so a kernel entry is not guaranteed.
4. (C) A page fault is a hardware exception. The CPU saves the state and enters kernel mode to run the page-fault handler. This happens every time, so it is guaranteed.
5. (D) A system call executes a trap instruction (a software interrupt). The CPU switches to kernel mode and runs the kernel service routine. This is guaranteed.
6. So (C) and (D) always cause the transition, while (A) never does and (B) only sometimes does.

KEY IDEA
The CPU enters kernel mode only through a trap, an interrupt or an exception. A system call and a page fault always do this. A plain function call never does, and a library call such as `malloc` does so only when it has to ask the kernel for more memory.

WATCH OUT FOR
Treating `malloc` as a system call. It is a library routine that enters the kernel (through `brk` or `mmap`) only when the heap must grow, so it does not guarantee a mode switch.

Q.24 MSQ · 1 MARK

Theory of Computation › Turing machines

Which of the following statements is/are CORRECT?

(A) The intersection of two regular languages is regular.

(B) The intersection of two context-free languages is context-free.

(C) The intersection of two recursive languages is recursive.

(D) The intersection of two recursively enumerable languages is recursively enumerable.

Official answer: **A, C, D**

SOLUTION

1. (A) Regular languages are closed under intersection. Build the product DFA of the two machines, and accept when both components accept. TRUE.
2. (B) Context-free languages are not closed under intersection. Take $L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$ and $L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$. Both are context-free. Their intersection is $\{a^n b^n c^n \mid n \geq 0\}$, which is not context-free by the pumping lemma for CFLs. FALSE.
3. (C) Recursive languages are closed under intersection. Run the deciders M_1 and M_2 one after the other on w, and accept only if both accept. Both always halt, so the new machine always halts. TRUE.
4. (D) Recursively enumerable languages are closed under intersection. Run M_1 and M_2 on w, one after the other or interleaved, and accept if both accept. If w is in both languages, both machines halt and accept. If not, the new machine need not halt, which is allowed for an RE language. TRUE.
5. Statements A, C and D are correct, and B is the only false one.

KEY IDEA
Regular, recursive and recursively enumerable languages are all closed under intersection. Context-free languages are not.

WATCH OUT FOR
Assuming context-free languages are closed under intersection because they are closed under union. The standard counterexample is $\{a^n b^n c^n\}$, which is the intersection of two context-free languages.

Q.25 MSQ · 1 MARK

Computer Networks › Routing

Which of the following statements is/are INCORRECT about the OSPF (Open Shortest Path First) routing protocol used in the Internet?

(A) OSPF implements Bellman-Ford algorithm to find shortest paths.

(B) OSPF uses Dijkstra's shortest path algorithm to implement least-cost path routing.

(C) OSPF is used as an inter-domain routing protocol.

(D) OSPF implements hierarchical routing.

Official answer: **A, C**

SOLUTION

1. OSPF is a link-state protocol. Each router floods link-state advertisements, so every router in an area builds the same topology database.
2. Each router then runs Dijkstra's shortest path algorithm on that database to compute least-cost routes. So (B) is a correct statement.
3. Bellman-Ford is the distance-vector algorithm, used by RIP. OSPF does not use it. So (A) is an INCORRECT statement.
4. OSPF is an interior gateway protocol (IGP) that runs inside one autonomous system. Routing between autonomous systems (inter-domain routing) is done by BGP. So (C) is an INCORRECT statement.
5. OSPF divides an autonomous system into areas joined through the backbone area 0. This is hierarchical routing. So (D) is a correct statement.
6. The question asks for the INCORRECT statements, which are (A) and (C).

KEY IDEA

OSPF is an interior, link-state protocol. It runs Dijkstra's algorithm on a link-state database and uses areas around a backbone for hierarchy. Bellman-Ford belongs to distance-vector protocols such as RIP, and inter-domain routing belongs to BGP.

WATCH OUT FOR

Picking the correct statements instead of the incorrect ones, because the question asks for INCORRECT. Another slip is confusing OSPF with BGP and calling it an inter-domain protocol.

Q.26 MSQ · 1 MARK

Discrete Mathematics › Mathematical logic

Geetha has a conjecture about integers, which is of the form

$$\forall x \left(P(x) \Rightarrow \exists y \, Q(x, y)\right),$$

where P is a statement about integers, and Q is a statement about pairs of integers. Which of the following (one or more) option(s) would imply Geetha's conjecture?

(A) $\exists x \left(P(x) \wedge \forall y \, Q(x, y)\right)$

(B) $\forall x \forall y \, Q(x, y)$

(C) $\exists y \forall x \left(P(x) \Rightarrow Q(x, y)\right)$

(D) $\exists x \left(P(x) \wedge \exists y \, Q(x, y)\right)$

Official answer: **B, C**

SOLUTION

1. The conjecture is $\forall x \left(P(x) \Rightarrow \exists y \, Q(x, y)\right)$. A statement implies it only if, whenever it is true, every x with $P(x)$ has some y with $Q(x, y)$.
2. (A) $\exists x \left(P(x) \wedge \forall y \, Q(x, y)\right)$ talks about one x only. Counter-model: take integers, $P(x)$ true for all x, and $Q(x, y)$ true exactly when $x = 0$. Then (A) holds with $x = 0$, but for $x = 1$ there is no y with $Q(1, y)$, so the conjecture fails. (A) does not imply it.
3. (B) $\forall x \, \forall y \, Q(x, y)$: fix any x. Since the integers are non-empty, take $y = 0$. Then $Q(x, 0)$ holds, so $\exists y \, Q(x, y)$ holds for every x. Hence $P(x) \Rightarrow \exists y \, Q(x, y)$ is true for every x. (B) implies the conjecture.

4. (C) $\exists y \forall x\,(P(x) \Rightarrow Q(x,y))$: fix a witness y_0 with $P(x) \Rightarrow Q(x,y_0)$ for all x. Take any x with $P(x)$. Then $Q(x,y_0)$ holds, so $\exists y\, Q(x,y)$. Hence $\forall x\,(P(x) \Rightarrow \exists y\, Q(x,y))$. This is the valid rule that $\exists y \forall x$ implies $\forall x \exists y$. (C) implies the conjecture.
5. (D) $\exists x\,(P(x) \wedge \exists y\, Q(x,y))$ again names only one x. Counter-model: $P(x)$ true for all x, $Q(x,y)$ true exactly when $x = 0$. (D) holds with $x = 0$, but $x = 1$ has no y with $Q(1,y)$, so the conjecture fails. (D) does not imply it.
6. Correct options: (B) and (C).

KEY IDEA
To imply a statement of the form $\forall x\,(P \Rightarrow \exists y\, Q)$, a premise must give a witness y for every x with $P(x)$. $\forall x \forall y$ and $\exists y \forall x$ both do. A premise about only one x (an $\exists x$ statement) cannot.

WATCH OUT FOR
Choosing (A) or (D) because they look like the conjecture, when an $\exists x$ premise says nothing about the other x. Also doubting (C) because its quantifier order differs: $\exists y \forall x$ is stronger than $\forall x \exists y$, so it implies it.

Q.27 MSQ · 1 MARK

Operating Systems › CPU scheduling

Which one or more of the following CPU scheduling algorithms can potentially cause starvation?

(A) First-in First-Out

(B) Round Robin

(C) Priority Scheduling

(D) Shortest Job First

Official answer: **A, C, D or C, D**

SOLUTION

1. Starvation means a ready process can wait indefinitely because other processes are always chosen ahead of it.
2. FIFO (FCFS): the ready queue is served strictly in arrival order. Every process reaches the front after a finite number of earlier arrivals, so no starvation.
3. Round Robin: each ready process gets a time quantum in cyclic order. With n ready processes a process waits at most $(n-1)q$ before its next turn, so no starvation.
4. Priority scheduling: the highest-priority ready process always runs. A steady stream of higher-priority arrivals can keep a low-priority process waiting forever, so starvation is possible (the usual cure is aging).
5. Shortest Job First (including its preemptive form SRTF): the shortest burst is always chosen. If short jobs keep arriving, a long job may never be picked, so starvation is possible.
6. Hence the algorithms that can cause starvation are Priority Scheduling and Shortest Job First, options C and D.

KEY IDEA
Starvation needs a scheduler that can keep choosing other processes ahead of a waiting one forever. Priority and SJF/SRTF do this. FIFO and Round Robin bound every process's waiting time.

WATCH OUT FOR
Thinking only priority scheduling starves, and forgetting that SJF/SRTF starves long jobs when short jobs keep arriving. Another slip is picking FIFO because a long job at the front delays others. That delay is finite, so it is not starvation.

Q.28 MSQ · 1 MARK

Engineering Mathematics › Calculus

Let

$$f(x) = x^3 + 15x^2 - 33x - 36$$

be a real-valued function.

Which of the following statements is/are TRUE?

(A) $f(x)$ does not have a local maximum.

(B) $f(x)$ has a local maximum.

(C) $f(x)$ does not have a local minimum.

(D) $f(x)$ has a local minimum.

Official answer: **B, D**

SOLUTION

1. A local extremum of a differentiable function occurs where $f'(x) = 0$. Differentiating, $f'(x) = 3x^2 + 30x - 33$.
2. Factor: $f'(x) = 3(x^2 + 10x - 11) = 3(x + 11)(x - 1)$, so the critical points are $x = -11$ and $x = 1$.
3. Second derivative: $f''(x) = 6x + 30$.
4. At $x = -11$: $f''(-11) = -66 + 30 = -36 < 0$, so f has a local maximum at $x = -11$.
5. At $x = 1$: $f''(1) = 6 + 30 = 36 > 0$, so f has a local minimum at $x = 1$.
6. Sign check: $f'(x) = 3(x + 11)(x - 1)$ is positive for $x < -11$, negative on $(-11, 1)$ and positive for $x > 1$. The sign changes from $+$ to $-$ at -11 (a maximum) and from $-$ to $+$ at 1 (a minimum).
7. So f has both a local maximum and a local minimum. Statements (B) and (D) are true; (A) and (C) are false.

KEY IDEA

Find the critical points from $f'(x) = 0$, then classify each with the sign of f'' (or the sign change of f'). A cubic whose derivative has two distinct real roots has one local maximum and one local minimum.

WATCH OUT FOR

Thinking a cubic cannot have local extrema because it is unbounded, or marking only one of the paired statements. Here the discriminant of f' is positive, so both extrema exist and both (B) and (D) must be chosen.

Q.29 MSQ · 1 MARK

Data Structures and Algorithms › Complexity

Let f and g be functions of natural numbers given by $f(n) = n$ and $g(n) = n^2$. Which of the following statements is/are TRUE?

(A) $f \in O(g)$

(B) $f \in \Omega(g)$

(C) $f \in o(g)$

(D) $f \in \Theta(g)$

Official answer: **A, C**

SOLUTION

1. **Definitions recap.** For $f(n) = n$ and $g(n) = n^2$: $f \in O(g)$ means $f(n) \leq c \cdot g(n)$ for some constant $c > 0$ and all sufficiently large n; $f \in \Omega(g)$ means $f(n) \geq c \cdot g(n)$ for some constant $c > 0$ and all sufficiently large n; $f \in o(g)$ means $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$; $f \in \Theta(g)$ means both $O(g)$ and $\Omega(g)$ hold.
2. **Option A ($f \in O(g)$):** Taking $c = 1$, $n \leq n^2$ for every $n \geq 1$. So $f(n) \leq c \cdot g(n)$ holds for all $n \geq 1$. TRUE.
3. **Option B ($f \in \Omega(g)$):** This would require $n \geq c \cdot n^2$ for some fixed constant $c > 0$ and all large n, i.e. $c \leq \frac{1}{n}$. But $\frac{1}{n} \to 0$ as $n \to \infty$, so no single positive constant c can satisfy this for all sufficiently large n.

FALSE.

4. **Option C ($f \in o(g)$):** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n}{n^2} = \lim_{n\to\infty} \frac{1}{n} = 0$. TRUE.
5. **Option D ($f \in \Theta(g)$):** $\Theta(g)$ requires both $O(g)$ AND $\Omega(g)$ to hold. Since $\Omega(g)$ was shown FALSE in Option B, $\Theta(g)$ cannot hold. FALSE.
6. **Conclusion:** The TRUE statements are A and C.

KEY IDEA

Compare growth rates of $f(n) = n$ and $g(n) = n^2$ using the formal definitions of O, Ω, o, and Θ: since n grows strictly slower than n^2, f is an upper-bounded-by (in fact strictly dominated, i.e. little-o) function of g, but not a lower bound, so Ω and Θ fail.

WATCH OUT FOR

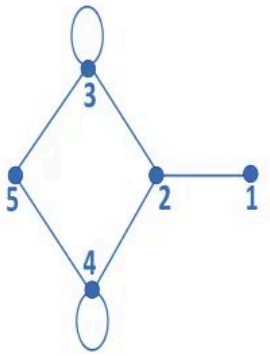
Assuming that because $f \in O(g)$ holds, $f \in \Theta(g)$ must also hold (treating O as if it already implied a tight bound), or forgetting that $o(g)$ (strict, limit-zero domination) is a stronger and still-true statement here, not something mutually exclusive with $O(g)$.

Q.30 NAT · 1 MARK

Engineering Mathematics › Linear algebra

Q.30

Let A be the adjacency matrix of the graph with vertices $\{1, 2, 3, 4, 5\}$.



Let λ_1, λ_2, λ_3, λ_4, and λ_5 be the five eigenvalues of A. Note that these eigenvalues need not be distinct.

The value of $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 =$ _____.

Official answer: **2, or 4**

SOLUTION

1. Read the graph from the figure. The edges are 1-2, 2-3, 2-4, 3-5 and 4-5. There is also a self-loop at vertex 3 and a self-loop at vertex 4.
2. In the adjacency matrix, $a_{ij} = 1$ when i and j are joined. A self-loop at i puts a 1 on the diagonal entry a_{ii}. A vertex with no loop has $a_{ii} = 0$.
3. So the diagonal is $(a_{11}, a_{22}, a_{33}, a_{44}, a_{55}) = (0, 0, 1, 1, 0)$.

4. The eigenvalues of a square matrix, counted with multiplicity, sum to its trace: $\sum_{i=1}^{5} \lambda_i = \operatorname{tr}(A)$. Repeated eigenvalues are counted each time, so the fact that they need not be distinct does not matter.
5. $\operatorname{tr}(A) = 0 + 0 + 1 + 1 + 0 = 2$, so $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 2$.
6. Some texts count a loop in an undirected graph as 2 on the diagonal. Then the trace would be 4. The official key accepts both 2 and 4, so the standard convention (loop gives 1) gives the answer 2.

KEY IDEA
The sum of the eigenvalues, counted with multiplicity, equals the trace of the matrix. You can read the trace straight off the diagonal without finding any eigenvalue. Here the diagonal entries come only from the self-loops.

WATCH OUT FOR
Trying to find the characteristic polynomial and the eigenvalues of the 5×5 matrix. Another slip is forgetting that a self-loop gives a diagonal entry, which makes the trace 0.

Q.31 NAT · 1 MARK

Engineering Mathematics › Calculus

The value of the definite integral

$$\int_{-3}^{3} \int_{-2}^{2} \int_{-1}^{1} \left(4x^2 y - z^3\right) \, \mathrm{d}z \, \mathrm{d}y \, \mathrm{d}x$$

is _____ . (Rounded off to the nearest integer)

Official answer: **0**

SOLUTION

1. The limits are constants, so the integral can be split term by term: $I = \int_{-3}^{3} \int_{-2}^{2} \int_{-1}^{1} 4x^2 y \, \mathrm{d}z \, \mathrm{d}y \, \mathrm{d}x - \int_{-3}^{3} \int_{-2}^{2} \int_{-1}^{1} z^3 \, \mathrm{d}z \, \mathrm{d}y \, \mathrm{d}x$.
2. First part: $4x^2 y$ does not depend on z, so the z-integral gives $\int_{-1}^{1} \mathrm{d}z = 2$. The first part becomes $\int_{-3}^{3} \int_{-2}^{2} 8x^2 y \, \mathrm{d}y \, \mathrm{d}x$.
3. In this double integral, $\int_{-2}^{2} y \, \mathrm{d}y = \left[\frac{y^2}{2}\right]_{-2}^{2} = 2 - 2 = 0$. The first part is therefore 0.
4. Second part: $\int_{-1}^{1} z^3 \, \mathrm{d}z = \left[\frac{z^4}{4}\right]_{-1}^{1} = \frac{1}{4} - \frac{1}{4} = 0$. The integrand is a product of functions of single variables, so the whole triple integral is 0 times the finite x and y integrals, which is 0.
5. Therefore $I = 0 - 0 = 0$.

KEY IDEA
An odd function integrated over an interval symmetric about zero gives zero. Each term of the integrand contains a factor that is odd in one variable (y in the first term, z^3 in the second), and the limits for that variable are symmetric.

WATCH OUT FOR
Computing every integral in full when symmetry settles it at once. Another slip is to treat the x^2 factor as the deciding one: it is even, so it does not make the first term vanish. The y factor does.

Q.32 NAT · 1 MARK

Digital Electronics › Number systems

A particular number is written as 132 in radix-4 representation. The same number in radix-5 representation is _____.

Official answer: **110**

SOLUTION

1. Write the number using positional weights in radix 4: $(132)_4 = 1 \cdot 4^2 + 3 \cdot 4^1 + 2 \cdot 4^0$.

2. $= 16 + 12 + 2 = 30$ in decimal.
3. Convert 30 to radix 5 by repeated division by 5: $30 \div 5 = 6$ remainder 0.
4. $6 \div 5 = 1$ remainder 1.
5. $1 \div 5 = 0$ remainder 1.
6. Read the remainders from last to first: 110.
7. Check: $1 \cdot 25 + 1 \cdot 5 + 0 \cdot 1 = 30$, so $(132)_4 = (110)_5$.

KEY IDEA
Convert through decimal: expand the radix-4 digits by powers of 4 to get 30, then divide repeatedly by 5 and read the remainders from last to first.

WATCH OUT FOR
Reading the remainders from first to last, which gives 011 instead of 110. Another slip is reporting the decimal value 30 and not the radix-5 digits.

Q.33 NAT · 1 MARK

Computer Organization and Architecture › Pipelining

Consider a 3-stage pipelined processor having a delay of 10 ns (nanoseconds), 20 ns, and 14 ns, for the first, second, and the third stages, respectively. Assume that there is no other delay and the processor does not suffer from any pipeline hazards. Also assume that one instruction is fetched every cycle.

The total execution time for executing 100 instructions on this processor is _____ ns.

Official answer: **2040**

SOLUTION

1. The processor is a k = 3 stage pipeline with per-stage delays 10 ns, 20 ns, and 14 ns.
2. With no hazards and a single shared pipeline clock, the cycle time must be long enough for the slowest stage: cycle time = max(10, 20, 14) = 20 ns.
3. One instruction is fetched (issued into the pipeline) every cycle, so for n = 100 instructions the pipeline needs k − 1 = 2 extra cycles to fill up plus n = 100 cycles for all instructions to be issued and drained, giving k + n − 1 = 3 + 100 − 1 = 102 cycles total.
4. Total execution time = 102 cycles × 20 ns/cycle = 2040 ns.

KEY IDEA
In a k-stage pipeline with no hazards, the cycle time is set by the slowest stage, and n instructions (one issued per cycle) finish in (k + n − 1) cycles: k−1 cycles to fill the pipeline plus n cycles for the n issues.

WATCH OUT FOR
Using the sum of all stage delays (10+20+14 = 44 ns) as the cycle time instead of the maximum stage delay (20 ns); a pipelined cycle time is bounded by the slowest single stage, not the total latency of one instruction.

Q.34 NAT · 1 MARK

Computer Organization and Architecture › I/O, interrupts & DMA

A keyboard connected to a computer is used at a rate of 1 keystroke per second. The computer system polls the keyboard every 10 ms (milli seconds) to check for a keystroke and consumes 100 µs (micro seconds) for each poll. If it is determined after polling that a key has been pressed, the system consumes an additional 200 µs to process the keystroke. Let T_1 denote the fraction of a second spent in polling and processing a keystroke.

In an alternative implementation, the system uses interrupts instead of polling. An interrupt is raised for every keystroke. It takes a total of 1 ms for servicing an interrupt and processing a keystroke. Let T_2 denote the fraction of a second spent in servicing the interrupt and processing a keystroke.

The ratio $\frac{T_1}{T_2}$ is _____. (Rounded off to one decimal place)

Official answer: **10.2**

SOLUTION

1. Since the keystroke rate is 1 per second, analyze a 1-second interval, in which exactly one keystroke occurs.
2. Polling: the system polls every 10 ms, so in 1 second (1000 ms) there are 1000/10 = 100 polls.
3. Each poll costs 100 µs, so total polling cost = 100 × 100 µs = 10,000 µs (= 10 ms).
4. Since one keystroke occurs in this second, an additional 200 µs is spent processing it once detected.
5. Total time spent on polling and processing in one second = 10,000 µs + 200 µs = 10,200 µs.
6. T1 = 10,200 µs / 1,000,000 µs (1 second in µs) = 0.0102.
7. Interrupt-driven: one interrupt occurs per second (matching the keystroke rate), and each interrupt costs a total of 1 ms = 1000 µs for servicing and processing.
8. T2 = 1000 µs / 1,000,000 µs = 0.001.
9. Ratio T1/T2 = 0.0102 / 0.001 = 10.2.

KEY IDEA

Compute the total time spent per second under each scheme (polling overhead + one keystroke-processing event per second, versus one interrupt-service event per second), express each as a fraction of a second, and take the ratio.

WATCH OUT FOR

Forgetting to add the 200 µs keystroke-processing time to the polling cost (giving T1 = 0.01, ratio 10.0), or forgetting that the 200 µs is added only once per second (since only one keystroke occurs per second), not per poll.

Q.35 NAT · 1 MARK

Programming in C › Functions & scope

The integer value printed by the ANSI-C program given below is _____.

```
#include<stdio.h>

int funcp(){
      static int x = 1;
      x++;
      return x;
}

int main(){
      int x,y;
      x = funcp();
      y = funcp()+x;
      printf("%d\n", (x+y));
      return 0;
}
```

Official answer: **7**

SOLUTION

1. funcp declares static int x = 1. A static local variable is initialized exactly once (at first call) and its value is retained across calls to funcp. main separately declares its own automatic int x, y, unrelated to funcp 's static x. They merely share a name.
2. First call: x = funcp(); . Inside funcp, x++ takes the static x from 1 to 2, and funcp returns 2. This is assigned to main 's x, so main 's x = 2.

3. Second call: `y = funcp() + x;`. Inside `funcp`, `x++` takes the static `x` from `2` to `3`, and `funcp` returns `3`. In this same statement, `main`'s `x` is only *read*, never modified, so there is no sequencing hazard: regardless of evaluation order, `main`'s `x` reads as its already-assigned value, `2`.
4. So `y = 3 + 2 = 5`.
5. `printf("%d\n", (x+y))` prints `main`'s `x + y = 2 + 5 = 7`.

KEY IDEA

A `static` local variable inside a function keeps its value across calls (initialized only once, at first entry) and is completely distinct from any same-named variable in the calling function; only reads (never a write) of a variable elsewhere in the same full expression are safe to reason about without worrying about C's unsequenced evaluation order.

WATCH OUT FOR

Confusing `funcp`'s static `x` with `main`'s local `x` and re-computing `main`'s `x` using the static variable's latest value (e.g. wrongly taking `main`'s `x` as `3` during the second call instead of the `2` it was already assigned), or assuming C's unspecified evaluation order makes `y = funcp() + x` undefined. It is not, because `x` is only read here, not modified, in this expression.

Q.36 MCQ · 2 MARKS

Programming in C › Recursion

Q.36 Consider the following program:

```
int main()
{
   f1();
   f2(2);
   f3();
   return(0);
}
```

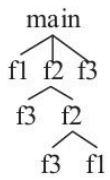
```
int f1()
{
  return(1);
}
```

```
int f2(int X)
{
   f3();
  if (X==1)
    return f1();
  else
    return (X*f2(X-1));
}
```

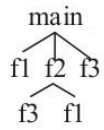
```
int f3()
{
  return(5);
}
```

Which one of the following options represents the activation tree corresponding to the main function?

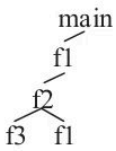
(A)



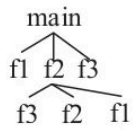
(B)



(C)



(D)



Official answer: **A**

SOLUTION

1. main() executes, in order: f1(); f2(2); f3(); return(0). So the root node main has three children, in this order: f1, f2, f3.
2. f1() only executes return(1); it makes no calls, so wherever f1 appears in the tree it is a leaf.
3. f3() only executes return(5); it makes no calls, so wherever f3 appears in the tree it is a leaf.

4. f2(2): the body of f2 first (unconditionally) calls f3(), a leaf child. Then it checks X==1; here X==2, so it takes the else branch: return (X*f2(X-1)), which calls f2(2-1) = f2(1). So the children of this f2 node, in call order, are f3 and f2(1).
5. f2(1): again first calls f3(), a leaf child. Then X==1 is true, so it executes return f1(), calling f1(). So the children of this f2(1) node, in call order, are f3 and f1.
6. Assembling the whole tree: main → {f1, f2, f3}; the f2 node just created → {f3, f2'}; that nested f2' node → {f3, f1}. This is a 3-level tree under the f2 branch, with main's other two children (f1, f3) remaining leaves.
7. Comparing to the options: (A) reproduces exactly this structure: main's three children f1,f2,f3; the f2 node's two children f3,f2; that inner f2 node's two children f3,f1. (B) shows only one extra level (f3,f1) directly under f2, omitting the intermediate recursive f2 node, so it is wrong: it collapses two calls into one. (C) makes f1 the sole child of main and puts f2 under f1, which is wrong: f1() is called and returns before f2(2) is even called; it is not f2's parent. (D) flattens f3, f2, f1 into one row of siblings under f2, which is wrong: it discards the fact that f2(1) is itself a call made by f2(2), with its own two children.
8. Therefore the correct option is (A).

KEY IDEA

An activation (call) tree mirrors the actual sequence of function calls made during execution, nested exactly as calls are made: if function P calls Q, Q becomes a child of P's node at that call site, in left-to-right call order. It is not the same as the static call graph, and repeated/recursive calls to the same function name create separate, deeper nodes rather than being merged or flattened.

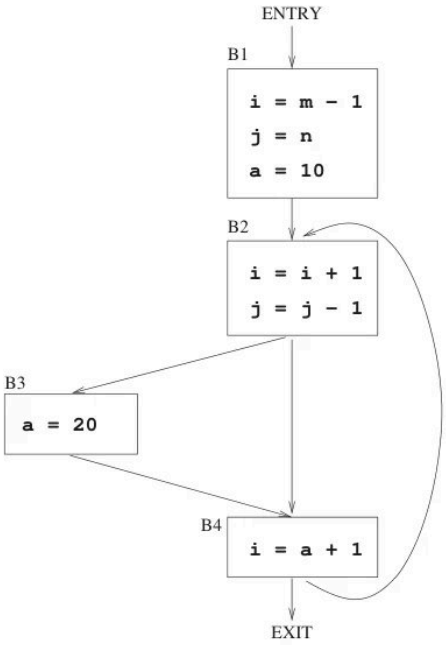
WATCH OUT FOR

Treating "f2 calls f3 then recursively calls f2" as if it collapses into one extra level (option B) or flattens all of f2's descendants into siblings of f2 (option D), instead of correctly nesting each recursive call one level deeper as its own node with its own children, losing track of exactly how many times f2 recurses (here once, from X=2 down to X=1) and forgetting that f3() is called again inside each new invocation of f2.

Q.37 MCQ · 2 MARKS

Compiler Design › Optimisation

Q.37 Consider the control flow graph shown.



Which one of the following choices correctly lists the set of *live* variables at the exit point of each basic block?

(A) B1: {}, B2: {a}, B3: {a}, B4: {a}

(B) B1: {i, j}, B2: {a}, B3: {a}, B4: {i}

(C) B1: {a, i, j}, B2: {a, i, j}, B3: {a, i}, B4: {a}

(D) B1: {a, i, j}, B2: {a, j}, B3: {a, j}, B4: {a, i, j}

Official answer: **D**

SOLUTION

1. A variable is live at a point if its current value may be used before it is redefined. Use the backward equations $\mathrm{IN}(B) = \mathrm{USE}(B) \cup (\mathrm{OUT}(B) - \mathrm{DEF}(B))$ and $\mathrm{OUT}(B) = \bigcup_{S \in \mathrm{succ}(B)} \mathrm{IN}(S)$.
2. Successors: B1 $\rightarrow$ B2; B2 $\rightarrow$ B3, B4; B3 $\rightarrow$ B4; B4 $\rightarrow$ B2 (back edge) and EXIT. Nothing is live at EXIT.
3. USE/DEF: B1 uses $\{m, n\}$ and defines $\{i, j, a\}$. B2 uses $\{i, j\}$ and defines $\{i, j\}$. B3 uses nothing and defines $\{a\}$. B4 uses $\{a\}$ and defines $\{i\}$. (m and n are never defined in the graph and do not appear

in the options.)

4. Because B2 uses i, j before redefining them, $\text{IN}(B2) = \{i, j\} \cup (\text{OUT}(B2) - \{i, j\})$. Also $\text{OUT}(B4) = \text{IN}(B2) \cup \varnothing = \text{IN}(B2)$, since the back edge feeds B2 and EXIT contributes nothing.
5. $\text{IN}(B4) = \{a\} \cup (\text{OUT}(B4) - \{i\})$. $\text{OUT}(B3) = \text{IN}(B4)$ and $\text{IN}(B3) = \text{OUT}(B3) - \{a\} \subseteq \text{IN}(B4)$. Hence $\text{OUT}(B2) = \text{IN}(B3) \cup \text{IN}(B4) = \text{IN}(B4)$.
6. Iterate from $\text{IN}(B2) = \{i, j\}$. Then $\text{OUT}(B4) = \{i, j\}$ and $\text{IN}(B4) = \{a\} \cup \{j\} = \{a, j\}$. So $\text{OUT}(B2) = \{a, j\}$ and $\text{IN}(B2) = \{i, j\} \cup (\{a, j\} - \{i, j\}) = \{a, i, j\}$. Recompute: $\text{OUT}(B4) = \{a, i, j\}$, so $\text{IN}(B4) = \{a\} \cup (\{a, i, j\} - \{i\}) = \{a, j\}$ again. Nothing changes, so this is the fixed point.
7. Exit sets: $\text{OUT}(B4) = \{a, i, j\}$; $\text{OUT}(B2) = \{a, j\}$; $\text{OUT}(B3) = \text{IN}(B4) = \{a, j\}$; $\text{OUT}(B1) = \text{IN}(B2) = \{a, i, j\}$.
8. This is B1: $\{a, i, j\}$, B2: $\{a, j\}$, B3: $\{a, j\}$, B4: $\{a, i, j\}$, which is option (D).

KEY IDEA

Liveness is a backward data-flow problem. Compute IN and OUT by iterating $\text{IN}(B) = \text{USE}(B) \cup (\text{OUT}(B) - \text{DEF}(B))$ to a fixed point, and remember that the loop's back edge feeds B4's exit into B2's entry.

WATCH OUT FOR

Forgetting the back edge B4 $\rightarrow$ B2, so that OUT(B4) is taken as empty or as just {a}. Another slip is dropping i from OUT(B4) because B4 redefines it. That is wrong, because i is live after B4 through B2, where it is used in `i = i + 1`.

Q.38 MCQ · 2 MARKS

Operating Systems › Synchronisation

Consider the two functions incr and decr shown below.

```
incr(){
wait(s);
X = X+1;
signal(s);
}
```

```
decr(){
wait(s);
X = X-1;
signal(s);
}
```

There are 5 threads each invoking incr once, and 3 threads each invoking decr once, on the same shared variable X. The initial value of X is 10.

Suppose there are two implementations of the semaphore s, as follows:

I-1: s is a binary semaphore initialized to 1.

I-2: s is a counting semaphore initialized to 2.

Let V1, V2 be the values of X at the end of execution of all the threads with implementations I-1, I-2, respectively.

Which one of the following choices corresponds to the minimum possible values of V1, V2, respectively?

(A) 15, 7

(B) 7, 7

(C) 12, 7

(D) 12, 8

Official answer: **C**

SOLUTION

1. **I-1 (binary semaphore, initial 1).** Only one thread can be between wait(s) and signal(s) at a time, so each update of X is atomic and no update is lost.
2. Every incr adds 1 and every decr subtracts 1, in any order, so $V_1 = 10 + 5 - 3 = 12$. It is the same in every execution, so the minimum is 12.
3. **I-2 (counting semaphore, initial 2).** Two threads can be inside the critical section together, so two read-modify-write sequences can overlap and an update can be overwritten (lost). A thread always writes the value it read, plus 1 (incr) or minus 1 (decr).
4. **Lower bound.** Let d be the number of decr writes completed so far. Claim: $X \geq 10 - d$ at all times. Initially $X = 10$ and $d = 0$. An incr writes $r + 1$, where r is an earlier value of X, and $r \geq 10 - d_{\text{then}} \geq 10 - d_{\text{now}}$, so the claim still holds. A decr writes $r - 1$ with $r \geq 10 - d_{\text{before}}$, and d rises by 1, so $X \geq 10 - d_{\text{after}}$. With only 3 decrements, $d \leq 3$, so the final value is at least $10 - 3 = 7$.
5. **A schedule that reaches 7.** Decr thread D_1 takes one permit, reads $X = 10$, and stalls inside the critical section.
6. The other permit is free for one thread at a time, so the 5 incr threads run one after another: $X = 10 \to 15$.
7. D_1 resumes and writes $10 - 1 = 9$, overwriting all five increments, so $X = 9$.
8. The two remaining decr threads run one after another: $9 \to 8 \to 7$. (If they overlapped and both read 9, the result would be 8, which is not smaller.)
9. So $V_2 = 7$ is reachable and cannot be beaten, and the minimum pair is $(V_1, V_2) = (12, 7)$, option (C).

KEY IDEA
A binary semaphore initialised to 1 gives mutual exclusion, so there are no lost updates. A counting semaphore initialised to 2 lets two threads overlap, so one thread's stale write can wipe out many increments. The value can never fall below $10 - 3 = 7$, because only decrements lower it, and each can lower it by at most 1 from some value that was itself at least that high.

WATCH OUT FOR
Taking 7 for V1 as well, as if the binary semaphore also allowed lost updates. Another slip is thinking the counting semaphore lets the decrements stack to $10 - 3 = 7$ only without losing increments, or forgetting that a stalled decr can overwrite the increments.

Q.39 MCQ · 2 MARKS

Theory of Computation › CFG & PDA

Consider the context-free grammar G below

$$S \to aSb \mid X$$
$$X \to aX \mid Xb \mid a \mid b$$

where S and X are non-terminals, and a and b are terminal symbols. The starting non-terminal is S.

Which one of the following statements is CORRECT?

(A) The language generated by G is $(a + b)^*$

(B) The language generated by G is $a^*(a + b)b^*$

(C) The language generated by G is $a^*b^*(a + b)$

(D) The language generated by G is not a regular language

Official answer: **B**

SOLUTION

1. Solve the inner non-terminal first. In $X \to aX \mid Xb \mid a \mid b$, each use of $X \to aX$ puts an a on the left and each use of $X \to Xb$ puts a b on the right. The derivation stops with exactly one terminal $c \in \{a, b\}$ from $X \to a$ or $X \to b$. So $L(X) = a^*(a+b)b^*$, that is, strings $a^i c\, b^j$ with $i, j \geq 0$ and $c \in \{a, b\}$.
2. $S \to aSb$ wraps $n \geq 0$ matching pairs around X, and $S \to X$ ends the wrapping. So $L(G) = \{a^n w\, b^n \mid n \geq 0,\ w \in a^*(a+b)b^*\}$.
3. Write $w = a^i c\, b^j$. Then the string is $a^{n+i}\, c\, b^{j+n}$, which has the form $a^m\, c\, b^k$ with $m, k \geq 0$. So $L(G) \subseteq a^*(a+b)b^*$.
4. Conversely, any string $a^m c\, b^k$ in $a^*(a+b)b^*$ is generated by taking $n = 0$, $i = m$, $j = k$, since $S \to X$ already gives all of $L(X)$. So $L(G) = a^*(a+b)b^*$, which is regular, and (B) is correct.
5. Check the others. (A): $ba \in (a+b)^*$, but $ba \notin L(G)$, because in $a^*(a+b)b^*$ every symbol after the single special symbol c is a b, and a string starting with b must have c as that first b, leaving only b's after it. So (A) is false. (C): $bba \in a^*b^*(a+b)$ by taking a^0b^2a, but $bba \notin L(G)$. The only symbols before c are a's, so c would be the first b with only b's after it, or the last a with only a's before it, and neither fits bba. So (C) is false. (D): $L(G)$ is regular, so (D) is false.

KEY IDEA

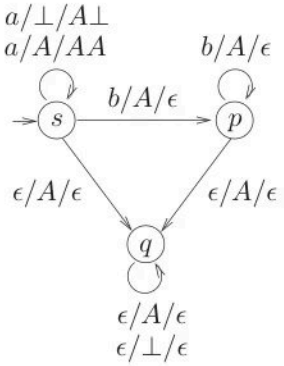
Find $L(X) = a^*(a+b)b^*$ first. The wrapping $a^n \cdots b^n$ in $S \to aSb$ cannot produce anything outside $a^*(a+b)b^*$, because the extra a's and b's fit into the existing starred parts. The language stays regular.

WATCH OUT FOR

Seeing $S \to aSb$ and assuming the language needs matched counts of a and b, hence non-regular (D). Here the middle part X already generates any number of a's before and any number of b's after the special symbol, so the matching pairs add nothing new.

Q.40 MCQ · 2 MARKS

Q.40 Consider the pushdown automaton (PDA) P below, which runs on the input alphabet $\{a, b\}$, has stack alphabet $\{\perp, A\}$, and has three states $\{s, p, q\}$, with s being the start state. A transition from state u to state v, labelled $c/X/\gamma$, where c is an input symbol or ϵ, X is a stack symbol, and γ is a string of stack symbols, represents the fact that in state u, the PDA can read c from the input, with X on the top of its stack, pop X from the stack, push in the string γ on the stack, and go to state v. In the initial configuration, the stack has only the symbol $\perp$ in it. The PDA accepts by empty stack.



Which one of the following options correctly describes the language accepted by P?

(A)	$\{a^m b^n \mid 1 \leq m \text{ and } n < m\}$
(B)	$\{a^m b^n \mid 0 \leq n \leq m\}$
(C)	$\{a^m b^n \mid 0 \leq m \text{ and } 0 \leq n\}$
(D)	$\{a^m \mid 0 \leq m\} \cup \{b^n \mid 0 \leq n\}$

Official answer: **A**

SOLUTION

1. In s with $\perp$ on top, the only move is $a/\perp/A\perp$, so the first symbol must be a. The empty string and any string starting with b are rejected, so $m \geq 1$. This rules out B, C and D, each of which accepts ε.
2. Reading m a's in s leaves the stack $A^m\perp$, because each $a/A/AA$ adds one A. No transition outside s reads a, so every a comes before every b.

3. Path 1: $s \to q$ by $\epsilon/A/\epsilon$ pops one A and reads no input. State q reads no input, so the remaining input must be empty, which gives $n = 0$. In q, the loops $\epsilon/A/\epsilon$ and $\epsilon/\perp/\epsilon$ pop the remaining A's and then $\perp$. So a^m with $m \geq 1$ is accepted.
4. Path 2: $s \to p$ by $b/A/\epsilon$ reads the first b and pops one A. In p, each further b pops one A (loop $b/A/\epsilon$). After $n \geq 1$ b's, n A's have been popped and $m - n$ remain.
5. To leave p we need the move $\epsilon/A/\epsilon$ ($p \to q$), which pops one more A. This needs $m - n \geq 1$, that is $n < m$. Then q pops the remaining A's and $\perp$, so the stack becomes empty.
6. If $n \geq m$, the A's run out before $p \to q$ can fire. $\perp$ is then on top, p has no move on $\perp$, and the stack cannot be emptied, so the string is rejected.
7. Combining the two paths, the language is $\{a^m b^n \mid m \geq 1,\ 0 \leq n < m\}$, which is option A.

KEY IDEA

Acceptance is by empty stack, and the only way to empty it is to get to q. Every b pops an A, and the move into q needs one more A plus input already used up. So at least one A must be left after the b's, which gives $n < m$.

WATCH OUT FOR

Choosing B ($n \leq m$) by thinking each b simply cancels an a. The exit from p to q pops one extra A, so $n = m$ strands $\perp$ on top. Also forgetting that ε is rejected, since the first move needs an a.

Q.41 MCQ · 2 MARKS

Computer Organization and Architecture › Instructions & addressing

Consider the given C-code and its corresponding assembly code, with a few operands U1–U4 being unknown. Some useful information as well as the semantics of each unique assembly instruction is annotated as inline comments in the code. The memory is byte-addressable.

```
 //C-code
int a[10], b[10], i;
// int is 32-bit
for (i=0; i<10;i++)
   a[i] = b[i] * 8;
```

```
 ;assembly-code (; indicates comments)
;r1-r5 are 32-bit integer registers
;initialize r1=0, r2=10
;initialize r3, r4 with base address of a, b
L01: jeq r1, r2, end  ;if(r1==r2) goto end
L02: lw   r5, 0(r4)   ;r5 ← Memory[r4+0]
L03: shl  r5, r5, U1  ;r5 ← r5 << U1
L04: sw   r5, 0(r3)   ;Memory[r3+0] ← r5
L05: add  r3, r3, U2  ;r3 ← r3+U2
L06: add  r4, r4, U3
L07: add  r1, r1, 1
L08: jmp  U4          ;goto U4
L09: end
```

Which one of the following options is a CORRECT replacement for operands in the position (U1, U2, U3, U4) in the above assembly code?

(A) (8, 4, 1, L02)

(B) (3, 4, 4, L01)

(C) (8, 1, 1, L02)

(D) (3, 1, 1, L01)

Official answer: **B**

SOLUTION

1. The loop is for(i=0;i<10;i++) a[i]=b[i]*8; with int = 4 bytes, byte-addressable memory.
2. L01 (jeq r1,r2,end) tests i==10; since it must run every iteration, it is the loop's back-edge target.
3. L02 loads b[i] via base register r4.
4. L03 must realize *8; since 8=2^3, the shift amount is U1 = 3 (not 8, which would give *256).
5. L04 stores the result into a[i] via base register r3.
6. L05 advances r3 (pointer into a) by one element; since int is 4 bytes and memory is byte-addressable, U2 = 4.
7. L06 advances r4 (pointer into b) the same way: U3 = 4.
8. L07 increments the loop counter i.
9. L08 (jmp U4) must return to the condition test, so U4 = L01 (jumping to L02 would skip the exit check).
10. Combining: (U1,U2,U3,U4) = (3,4,4,L01), which is option (B).

KEY IDEA

Translate the C loop into the assembly skeleton line by line: the shift amount for *8 is log2(8)=3, and pointer increments in a byte-addressable memory must equal the element size (sizeof(int) = 4 bytes here), not 1.

WATCH OUT FOR

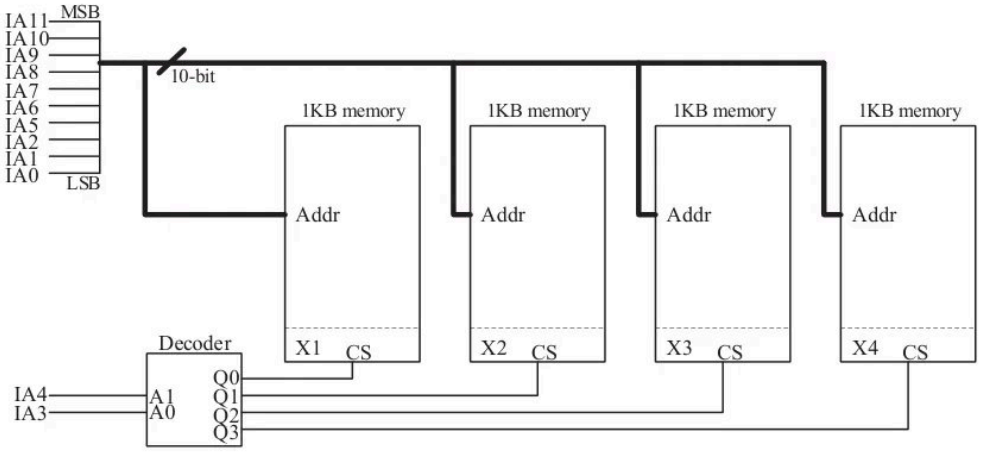
Picking U1 = 8 (using the multiplier itself as the shift amount) instead of log2(8) = 3, or forgetting that int is 4 bytes so the byte-addressable pointer increments (U2, U3) must be 4, not 1.

Q.42 MCQ · 2 MARKS

Computer Organization and Architecture › Memory & cache

Q.42

A 4 kilobyte (KB) byte-addressable memory is realized using four 1 KB memory blocks. Two input address lines (IA4 and IA3) are connected to the chip select (CS) port of these memory blocks through a decoder as shown in the figure. The remaining ten input address lines from IA11–IA0 are connected to the address port of these blocks. The chip select (CS) is active high.



The input memory addresses (IA11–IA0), in decimal, for the starting locations (Addr=0) of each block (indicated as X1, X2, X3, X4 in the figure) are among the options given below. Which one of the following options is CORRECT?

(A)	(0, 1, 2, 3)
(B)	(0, 1024, 2048, 3072)
(C)	(0, 8, 16, 24)
(D)	(0, 0, 0, 0)

Official answer: **C**

SOLUTION

1. Total memory = 4KB = 2^12 bytes, so 12 address lines IA11–IA0 are needed, matching the given lines.
2. Each 1KB block needs 10 address bits for its Addr port (2^10 = 1024 bytes). The figure shows exactly 10 of the 12 lines (all except IA4 and IA3) feeding the Addr port of every block in parallel.
3. IA4 and IA3 are bits 4 and 3 of the address (not the MSBs). They instead feed the 2-to-4 decoder as A1=IA4, A0=IA3, whose outputs Q0..Q3 chip-select X1..X4 respectively.

4. At a block's starting location (Addr=0), all 10 bits that feed the Addr port (bits 11,10,9,8,7,6,5,2,1,0) are 0 by definition.
5. X1 selected when Q0 is active, i.e. IA4 IA3 = 00 ⇒ bit4=0, bit3=0. With all other bits 0, the 12-bit address IA11..IA0 = 0 ⇒ decimal 0.
6. X2 selected when Q1 is active, i.e. IA4 IA3 = 01 ⇒ bit4=0, bit3=1, all other bits 0 ⇒ address = $2^3 = 8$.
7. X3 selected when Q2 is active, i.e. IA4 IA3 = 10 ⇒ bit4=1, bit3=0, all other bits 0 ⇒ address = $2^4 = 16$.
8. X4 selected when Q3 is active, i.e. IA4 IA3 = 11 ⇒ bit4=1, bit3=1, all other bits 0 ⇒ address = $2^4 + 2^3 = 24$.
9. So (X1,X2,X3,X4) = (0, 8, 16, 24), matching option (C).

KEY IDEA

The 10-bit Addr port of every block is fed by all address lines EXCEPT IA4 and IA3, which instead go to a 2-to-4 decoder that generates the chip-select for each block. So "starting location" (Addr port = 0) only forces the ten bus bits to 0; IA4 and IA3 are free to vary block-to-block since they only pick which block is selected, and their contribution to the 12-bit address is 2^4 and 2^3, not the top two (MSB) bits.

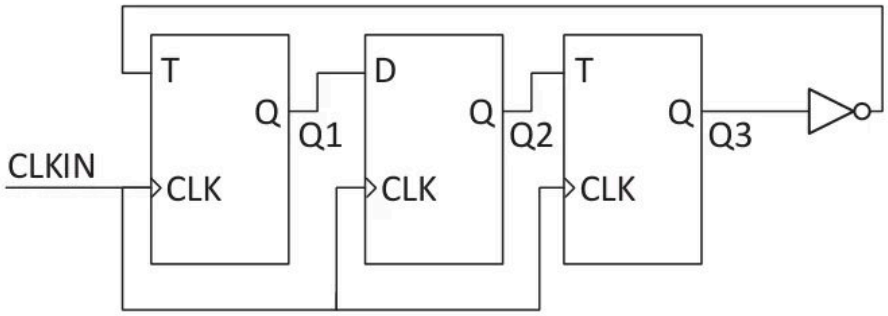
WATCH OUT FOR

Assuming the two lines going to the decoder must be the two MSBs (IA11, IA10), which would wrongly give offsets in multiples of 1024 (option B), or assuming a decoder always implies a simple linear block order 0,1,2,3 (option A). The trap is that IA4 and IA3 sit in the middle of the address, not at either end, so their weight is $2^4 = 16$ and $2^3 = 8$.

Q.43 MCQ · 2 MARKS

Digital Electronics › Sequential

Q.43 Consider a sequential digital circuit consisting of T flip-flops and D flip-flops as shown in the figure. CLKIN is the clock input to the circuit. At the beginning, Q1, Q2 and Q3 have values 0, 1 and 1, respectively.



Which one of the given values of (Q1, Q2, Q3) can NEVER be obtained with this digital circuit?

(A)	(0, 0, 1)
(B)	(1, 0, 0)
(C)	(1, 0, 1)
(D)	(1, 1, 1)

Official answer: **A**

SOLUTION

1. Read the connections: flip-flop 1 is a T flip-flop with $T_1 = \overline{Q_3}$ (the inverter output returns to its T input), flip-flop 2 is a D flip-flop with $D_2 = Q_1$, and flip-flop 3 is a T flip-flop with $T_3 = Q_2$. All three share CLKIN, so they change together on each clock edge.
2. A T flip-flop gives $Q^+ = T \oplus Q$ and a D flip-flop gives $Q^+ = D$. So $Q_1^+ = Q_1 \oplus \overline{Q_3}$, $Q_2^+ = Q_1$, $Q_3^+ = Q_3 \oplus Q_2$.
3. Start at $(Q_1, Q_2, Q_3) = (0, 1, 1)$: $Q_1^+ = 0 \oplus 0 = 0$, $Q_2^+ = 0$, $Q_3^+ = 1 \oplus 1 = 0$, giving $(0, 0, 0)$.

4. From $(0,0,0)$: $Q_1^+ = 0 \oplus 1 = 1$, $Q_2^+ = 0$, $Q_3^+ = 0 \oplus 0 = 0$, giving $(1,0,0)$. From $(1,0,0)$: $Q_1^+ = 1 \oplus 1 = 0$, $Q_2^+ = 1$, $Q_3^+ = 0 \oplus 0 = 0$, giving $(0,1,0)$.
5. From $(0,1,0)$: $Q_1^+ = 0 \oplus 1 = 1$, $Q_2^+ = 0$, $Q_3^+ = 0 \oplus 1 = 1$, giving $(1,0,1)$. From $(1,0,1)$: $Q_1^+ = 1 \oplus 0 = 1$, $Q_2^+ = 1$, $Q_3^+ = 1 \oplus 0 = 1$, giving $(1,1,1)$.
6. From $(1,1,1)$: $Q_1^+ = 1 \oplus 0 = 1$, $Q_2^+ = 1$, $Q_3^+ = 1 \oplus 1 = 0$, giving $(1,1,0)$. From $(1,1,0)$: $Q_1^+ = 1 \oplus 1 = 0$, $Q_2^+ = 1$, $Q_3^+ = 0 \oplus 1 = 1$, giving $(0,1,1)$, the start.
7. The circuit cycles through $011 \to 000 \to 100 \to 010 \to 101 \to 111 \to 110 \to 011$, seven states, and then repeats. Options B, C and D, namely $(1,0,0)$, $(1,0,1)$ and $(1,1,1)$, all appear in this cycle. The state 001 is not in the cycle, so it can never be reached from the start. Option (A), $(0,0,1)$, can never occur.

KEY IDEA

Write the next-state equations from the figure ($Q^+ = T \oplus Q$ for a T flip-flop, $Q^+ = D$ for a D flip-flop), then follow the circuit clock by clock from the given start state. Only the states on that path can ever occur.

WATCH OUT FOR

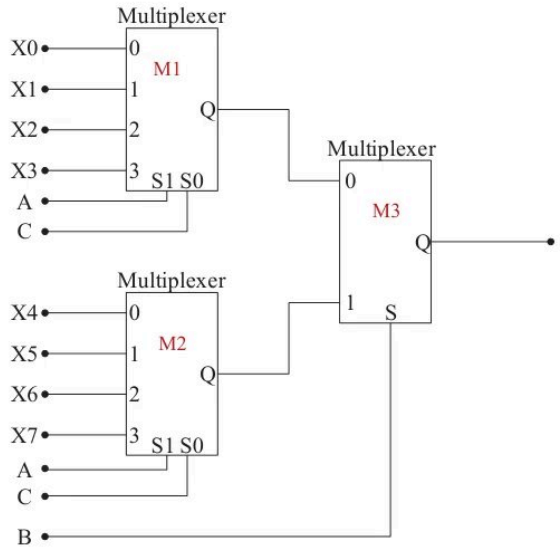
Forgetting the inverter on the feedback to the first T input (using $T_1 = Q_3$ instead of $\overline{Q_3}$), or testing all eight states instead of only those reachable from the start state $(0,1,1)$.

Q.44 MCQ · 2 MARKS

Digital Electronics › Combinational

Q.44

A Boolean digital circuit is composed using two 4-input multiplexers (M1 and M2) and one 2-input multiplexer (M3) as shown in the figure. X0–X7 are the inputs of the multiplexers M1 and M2 and could be connected to either 0 or 1. The select lines of the multiplexers are connected to Boolean variables A, B and C as shown.



Which one of the following set of values of (X0, X1, X2, X3, X4, X5, X6, X7) will realise the Boolean function $\overline{A} + \overline{A}.\overline{C} + A.\overline{B}.C$?

(A) (1, 1, 0, 0, 1, 1, 1, 0)

(B) (1, 1, 0, 0, 1, 1, 0, 1)

(C) (1, 1, 0, 1, 1, 1, 0, 0)

(D) (0, 0, 1, 1, 0, 1, 1, 1)

Official answer: **C**

SOLUTION

1. A 4-to-1 multiplexer with selects S_1S_0 outputs the data input numbered $2S_1 + S_0$. With $S_1 = A$ and $S_0 = C$, M1 outputs X_{2A+C} and M2 outputs X_{4+2A+C}.
2. M3 has select B: for $B = 0$ the output is M1's, and for $B = 1$ it is M2's.
3. Simplify the target: $\overline{A} + \overline{A}\,\overline{C} = \overline{A}$ (absorption), so $F = \overline{A} + A\overline{B}C$. By the rule $X + \overline{X}Y = X + Y$, this gives $F = \overline{A} + \overline{B}C$.

4. Case $B = 0$ (M1 selected): $\overline{B}C = C$, so $F = \overline{A} + C$.
5.

$A\,C$	F	input
00	1	X_0
01	1	X_1
10	0	X_2
11	1	X_3

So $(X_0, X_1, X_2, X_3) = (1, 1, 0, 1)$.

6. Case $B = 1$ (M2 selected): $\overline{B}C = 0$, so $F = \overline{A}$.
7.

$A\,C$	F	input
00	1	X_4
01	1	X_5
10	0	X_6
11	0	X_7

So $(X_4, X_5, X_6, X_7) = (1, 1, 0, 0)$.

8. Together $(X_0, \ldots, X_7) = (1, 1, 0, 1, 1, 1, 0, 0)$, which is option (C).

KEY IDEA
Treat the 2-input multiplexer as a Shannon expansion on B. Each 4-input multiplexer is a truth-table lookup on (A, C), so every data input is the value of F for one (B, A, C) combination.

WATCH OUT FOR
Swapping the select order ($S_1 = C$, $S_0 = A$), which reorders X_1 and X_2. Another slip is not simplifying $\overline{A} + A\overline{B}C$, which hides that F reduces to $\overline{A}$ when $B = 1$.

Q.45 MCQ · 2 MARKS

Digital Electronics › Number systems

Consider the IEEE-754 single precision floating point numbers P= `0xC1800000` and Q= `0x3F5C2EF4`. Which one of the following corresponds to the product of these numbers (i.e., $P \times Q$), represented in the IEEE-754 single precision format?

(A) `0x404C2EF4`

(B) `0x405C2EF4`

(C) `0xC15C2EF4`

(D) `0xC14C2EF4`

Official answer: **C**

SOLUTION

1. IEEE 754 single precision: $(-1)^s \times 1.f \times 2^{e-127}$, with 1 sign bit, 8 exponent bits and 23 fraction bits.
2. P = `0xC1800000` = $1100\,0001\,1000\,0000\ldots$: sign 1, exponent bits $10000011 = 131$, fraction 0. So $P = -1.0 \times 2^{131-127} = -16$.
3. Q = `0x3F5C2EF4` = $0011\,1111\,0101\,1100\,0010\,1110\,1111\,0100$: sign 0, exponent bits $01111110 = 126$, fraction bits $101\,1100\,0010\,1110\,1111\,0100$ (that is `0x5C2EF4`). So $Q = 1.f_Q \times 2^{126-127} = 1.f_Q \times 2^{-1}$.
4. Multiply: $P \times Q = -1.0 \times 2^4 \times 1.f_Q \times 2^{-1} = -1.f_Q \times 2^3$. Since the significand of P is exactly 1.0, the product significand is $1.f_Q$, which is already normalised, so the fraction stays `0x5C2EF4` and no rounding is needed.
5. Result: sign $= 1$, exponent $= 3 + 127 = 130 = 10000010$, fraction $= 101\,1100\,0010\,1110\,1111\,0100$.
6. Assemble the bits: $1\ 10000010\ 101\,1100\,0010\,1110\,1111\,0100 = 1100\,0001\,0101\,1100\,0010\,1110\,1111\,0100$, which is `0xC15C2EF4`. This is option (C).

KEY IDEA

P is exactly -1.0×2^4, so multiplying by it only flips the sign and adds 4 to the exponent of Q. The fraction field is unchanged and there is no rounding: exponent $126 + 4 = 130$, sign 1, fraction `0x5C2EF4`.

WATCH OUT FOR

Forgetting that the stored exponent is biased: adding the two biased exponents (131 + 126) without subtracting 127 once. Another slip is mis-aligning the 9 leading bits (sign plus exponent) when regrouping into hex, which gives `0x40..` or `0xC14C..` in place of `0xC15C..`.

Q.46 MCQ · 2 MARKS

Data Structures and Algorithms › Trees & heaps

Let A be a priority queue for maintaining a set of elements. Suppose A is implemented using a max-heap data structure. The operation $\text{EXTRACT-MAX}(A)$ extracts and deletes the maximum element from A. The operation $\text{INSERT}(A, key)$ inserts a new element key in A. The properties of a max-heap are preserved at the end of each of these operations.

When A contains n elements, which one of the following statements about the worst case running time of these two operations is TRUE?

(A) Both $\text{EXTRACT-MAX}(A)$ and $\text{INSERT}(A, key)$ run in $O(1)$.

(B) Both $\text{EXTRACT-MAX}(A)$ and $\text{INSERT}(A, key)$ run in $O(\log(n))$.

(C) $\text{EXTRACT-MAX}(A)$ runs in $O(1)$ whereas $\text{INSERT}(A, key)$ runs in $O(n)$.

(D) $\text{EXTRACT-MAX}(A)$ runs in $O(1)$ whereas $\text{INSERT}(A, key)$ runs in $O(\log(n))$.

Official answer: **B**

SOLUTION

1. A max-heap on n elements is stored as a nearly complete binary tree of height $\Theta(\log(n))$, kept in an array.
2. $\text{INSERT}(A, key)$: place key at the next free array position (a new leaf), then sift it up: repeatedly compare with its parent and swap while the max-heap property is violated. In the worst case the element travels from a leaf all the way to the root, giving $O(\log(n))$ comparisons and swaps.
3. $\text{EXTRACT-MAX}(A)$: the maximum sits at the root, so reading it is $O(1)$. To remove it while keeping the tree complete, the LAST element in the array is moved into the root position and the heap size is decremented.
4. This relocated element is then sifted down (max-heapify): at each level it is compared with its larger child and swapped down if it violates the max-heap property, following a single root-to-leaf path.
5. Sifting down takes $O(\log(n))$ comparisons and swaps in the worst case, since it can traverse the full height of the tree.
6. So $\text{EXTRACT-MAX}(A)$ costs $O(1)$ (locate the max) plus $O(\log(n))$ (restore the heap) $= O(\log(n))$ overall, the same asymptotic order as $\text{INSERT}(A, key)$.
7. Hence both operations run in $O(\log(n))$ worst-case time, so statement (B) is TRUE.

KEY IDEA

A binary (max-)heap stored as a nearly complete binary tree has height $\Theta(\log(n))$; both INSERT (sift-up from a new leaf) and EXTRACT-MAX (move last element to root, then sift-down/max-heapify) do work proportional to the tree's height, so both are $O(\log(n))$ in the worst case. Reading the max itself is $O(1)$, but restoring the heap property after removing it is not.

WATCH OUT FOR

Assuming EXTRACT-MAX is $O(1)$ overall because "the maximum is right at the root" (options C/D). This only covers locating/reading the max. Removing it still requires refilling the root (with the last array element) and sifting it down through up to $\Theta(\log(n))$ levels to restore the max-heap property, so the operation as a whole costs $O(\log(n))$, not $O(1)$.

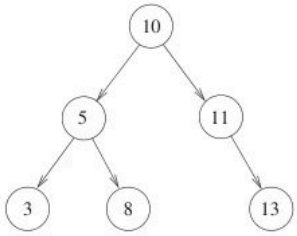
Q.47 MCQ · 2 MARKS

Programming in C › Recursion

Q.47 Consider the C function `foo` and the binary tree shown.

```
typedef struct node {
  int val;
  struct node *left, *right;
} node;

int foo(node *p) {
  int retval;
  if (p == NULL)
    return 0;
  else {
    retval = p->val + foo(p->left) + foo(p->right);
    printf("%d ", retval);
    return retval;
  }
}
```



When `foo` is called with a pointer to the root node of the given binary tree, what will it print?

(A) 3 8 5 13 11 10

(B) 3 5 8 10 11 13

(C) 3 8 16 13 24 50

(D) 3 16 8 50 24 13

Official answer: **C**

SOLUTION

1. The printf executes only after BOTH recursive calls foo(p→left) and foo(p→right) have returned and retval has been fully computed, so the print order follows post-order traversal (left subtree, then right subtree, then the current node), and each printed value is the SUM of that node's entire subtree, not the node's own stored value.
2. Leaf node 3 (left child of 5): retval = 3 + foo(NULL) + foo(NULL) = 3 + 0 + 0 = 3. Prints "3".

3. Leaf node 8 (right child of 5): retval = 8 + 0 + 0 = 8. Prints "8".
4. Node 5: foo(left = 3) runs first and prints "3"; then foo(right = 8) runs and prints "8"; then retval = 5 + 3 + 8 = 16. Prints "16". Output so far: 3 8 16.
5. Leaf node 13 (right child of 11; 11 has no left child): retval = 13 + foo(NULL) + foo(NULL) = 13 + 0 + 0 = 13. Prints "13".
6. Node 11: foo(left = NULL) returns 0 with no print (base case just returns, no printf runs when p == NULL); then foo(right = 13) runs and prints "13"; then retval = 11 + 0 + 13 = 24. Prints "24". Output so far: 3 8 16 13 24.
7. Node 10 (root): foo(left = 5) runs to completion first, producing prints 3, 8, 16; then foo(right = 11) runs to completion, producing prints 13, 24; then retval = 10 + 16 + 24 = 50. Prints "50".
8. Complete printed sequence: 3 8 16 13 24 50. This is option (C).

KEY IDEA

Each call to `foo(p)` first recurses into both children (left, then right) and only after both have returned does it compute `retval = p->val + foo(left) + foo(right)` and print it, so this is a post-order traversal, but what gets printed at each node is the sum of the entire subtree rooted there, not the node's own stored value.

WATCH OUT FOR

Assuming the printf prints the node's own value at the moment of visiting it (as in a plain post-order traversal print), giving 3 8 5 13 11 10 (option A), instead of recognizing that `retval` accumulates the subtree sum before printing, since the printf statement executes only after both recursive calls have already returned and `retval` has already been updated.

Q.48 MCQ · 2 MARKS

Discrete Mathematics › Counting

Let $U = \{1, 2, \ldots, n\}$, where n is a large positive integer greater than 1000. Let k be a positive integer less than n. Let A, B be subsets of U with $|A| = |B| = k$ and $A \cap B = \emptyset$. We say that a permutation of U separates A from B if one of the following is true.

- All members of A appear in the permutation before any of the members of B.
- All members of B appear in the permutation before any of the members of A.

How many permutations of U separate A from B?

(A) $n!$

(B) $\binom{n}{2k}(n-2k)!$

(C) $\binom{n}{2k}(n-2k)!\,(k!)^2$

(D) $2\binom{n}{2k}(n-2k)!\,(k!)^2$

Official answer: **D**

SOLUTION

1. Count the permutations in which every member of A comes before every member of B, then double the count for the symmetric case in which B comes first.
2. Choose which $2k$ of the n positions hold the members of $A \cup B$. This can be done in $\binom{n}{2k}$ ways.
3. Read those $2k$ positions from left to right. For A to come entirely before B, the first k of them must hold A and the last k must hold B. This is forced, so it adds no extra choice.
4. Arrange the k members of A in their k positions in $k!$ ways. Arrange the k members of B in their k positions in $k!$ ways.
5. Arrange the remaining $n - 2k$ elements of U in the remaining $n - 2k$ positions in $(n-2k)!$ ways.
6. The number of permutations with A entirely before B is $\binom{n}{2k}\,(n-2k)!\,(k!)^2$.
7. The number with B entirely before A is the same, by symmetry. The two cases cannot overlap: A and B are disjoint and non-empty, so a permutation cannot have both A before B and B before A.

8. Total: $2\binom{n}{2k}\,(n-2k)!\,(k!)^2$, which is option (D).
9. Check on small cases. For $n=2$, $k=1$: the formula gives $2\cdot\binom{2}{2}\cdot 0!\cdot 1=2$, and both permutations of $\{a,b\}$ separate. For $n=3$, $k=1$: the formula gives $2\cdot 3\cdot 1\cdot 1=6=3!$, and every permutation separates two single elements.

KEY IDEA
Pick the positions of $A\cup B$, and the order within them is forced (the first k for one set, the last k for the other). Then arrange each group and the rest, and multiply by 2 for the two possible orders of the sets.

WATCH OUT FOR
Forgetting the factor 2 for B appearing before A gives option (C). Forgetting the internal orderings $(k!)^2$ gives option (B).

Q.49 MSQ · 2 MARKS

Discrete Mathematics › Sets & relations

Let $f: A \to B$ be an onto (or surjective) function, where A and B are nonempty sets. Define an equivalence relation $\sim$ on the set A as

$$a_1 \sim a_2 \text{ if } f(a_1) = f(a_2),$$

where $a_1, a_2 \in A$. Let $\mathcal{E} = \{[x] : x \in A\}$ be the set of all the equivalence classes under $\sim$. Define a new mapping $F: \mathcal{E} \to B$ as

$$F([x]) = f(x), \text{ for all the equivalence classes } [x] \text{ in } \mathcal{E}.$$

Which of the following statements is/are TRUE?

(A) F is NOT well-defined.

(B) F is an onto (or surjective) function.

(C) F is a one-to-one (or injective) function.

(D) F is a bijective function.

Official answer: **B, C, D**

SOLUTION

1. **Well-defined.** F is defined on a class $[x]$ through one chosen representative x. If $[x]=[y]$, then $x\sim y$, so $f(x)=f(y)$ by the definition of $\sim$. The value $F([x])=f(x)$ therefore does not depend on the representative, so F is well-defined. Option (A) is false.
2. **Onto.** Take any $b\in B$. Since f is onto, there is $a\in A$ with $f(a)=b$. Then $F([a])=f(a)=b$, so every element of B has a preimage under F. Option (B) is true.
3. **One-to-one.** Suppose $F([x])=F([y])$. Then $f(x)=f(y)$, which by the definition of $\sim$ means $x\sim y$, so $[x]=[y]$. Hence F is injective. Option (C) is true.
4. **Bijective.** A bijective function is both onto and one-to-one. Steps 2 and 3 show both, so F is bijective. Option (D) is true.
5. **Conclusion.** The true statements are (B), (C) and (D). In other words, the classes of $\sim$ correspond one-to-one with the elements of B: $A/\!\sim\ \cong B$ whenever f is onto.

KEY IDEA
Grouping the elements of A that f sends to the same value gives a map from classes to B that is well-defined by the definition of $\sim$. The map is injective because equal values force equal classes, and onto because f is onto. So it is bijective.

WATCH OUT FOR
Picking only (B) and (C), or forgetting that (D) follows from them. Another slip is doubting well-definedness: the check is that the value does not depend on which representative of the class is used, and here that holds by the definition of $\sim$.

Q.50 MSQ · 2 MARKS

Computer Networks › TCP

Suppose you are asked to design a new reliable byte-stream transport protocol like TCP. This protocol, named `myTCP`, runs over a 100 Mbps network with Round Trip Time of 150 milliseconds and the maximum segment lifetime of 2 minutes.

Which of the following is/are valid lengths of the **Sequence Number** field in the `myTCP` header?

(A) 30 bits

(B) 32 bits

(C) 34 bits

(D) 36 bits

Official answer: **B, C, D**

SOLUTION

1. A byte-stream protocol numbers every byte. A sequence number must not be reused while an old segment carrying it can still be alive in the network, so the sequence space must not wrap around within the maximum segment lifetime (MSL).
2. Link rate: $100\ \text{Mbps} = 12.5 \times 10^6$ bytes/s. MSL $= 2\ \text{min} = 120\ \text{s}$.
3. Bytes that can be sent in one MSL: $12.5 \times 10^6 \times 120 = 1.5 \times 10^9$ bytes. The sequence space must be at least this large, so $2^n \geq 1.5 \times 10^9$. Since $2^{30} \approx 1.07 \times 10^9 < 1.5 \times 10^9 \leq 2^{31} \approx 2.15 \times 10^9$, at least 31 bits are needed.
4. A more conservative design allows for a stale segment living one MSL and its acknowledgement another MSL, so the numbers must stay unique over $2\times$MSL: $2 \times 1.5 \times 10^9 = 3 \times 10^9$ bytes. Then $2^{31} \approx 2.15 \times 10^9 < 3 \times 10^9 \leq 2^{32} \approx 4.29 \times 10^9$, so at least 32 bits are needed.
5. Window check: the bandwidth-delay product is $12.5 \times 10^6 \times 0.15 \approx 1.9 \times 10^6$ bytes, far below 2^{n-1} for $n \geq 30$, so the window size does not restrict the choice.
6. 30 bits gives only $2^{30} \approx 1.07 \times 10^9 < 1.5 \times 10^9$ bytes, so the numbers wrap within a single MSL under either reading. Option A is invalid.
7. 32, 34 and 36 bits each give $2^n \geq 2^{32} \approx 4.29 \times 10^9 \geq 3 \times 10^9$, so they satisfy even the conservative requirement. A longer field only enlarges the sequence space and never causes reuse within the MSL, so these lengths are all valid. Options B, C and D are correct.

KEY IDEA

The sequence number space must not wrap around within the maximum segment lifetime: bytes sent in one MSL are $12.5 \times 10^6 \times 120 = 1.5 \times 10^9$, so the field needs more than 30 bits, and any length of 32 bits or more is safe.

WATCH OUT FOR

Counting bits from the bandwidth-delay product (about 1.9×10^6 bytes, roughly 21 bits) instead of the bytes sent in one MSL, or treating only the smallest length as valid and ignoring that longer fields are also valid.

Q.51 MSQ · 2 MARKS

Discrete Mathematics › Groups

Let X be a set and 2^X denote the powerset of X. Define a binary operation Δ on 2^X as follows:

$$A \Delta B = (A - B) \cup (B - A).$$

Let $H = (2^X, \Delta)$. Which of the following statements about H is/are correct?

(A) H is a group.

(B) Every element in H has an inverse, but H is NOT a group.

(C) For every $A \in 2^X$, the inverse of A is the complement of A.

(D) For every $A \in 2^X$, the inverse of A is A.

Official answer: **A, D**

SOLUTION

1. Closure: $A\Delta B = (A - B) \cup (B - A)$ is built from elements of X, so it is a subset of X and lies in 2^X.
2. Associativity: an element x lies in $A\Delta B$ exactly when it belongs to exactly one of A, B, that is, to an odd number of them. For three sets, $x \in (A\Delta B)\Delta C$ exactly when the number of sets among $A\Delta B$ and C containing x is odd. This happens exactly when x lies in an odd number of A, B, C: if x is in C, it must lie in an even number of A, B; if not, in an odd number. The same count describes $A\Delta(B\Delta C)$. Both sides contain exactly the elements that belong to an odd number of A, B, C, so Δ is associative.
3. Identity: $A\Delta\varnothing = (A - \varnothing) \cup (\varnothing - A) = A \cup \varnothing = A$, and $\varnothing\Delta A = A$ by symmetry. So $\varnothing$ is the identity.
4. Inverse: $A\Delta A = (A - A) \cup (A - A) = \varnothing$. Every element is its own inverse, so (D) is true.
5. Closure, associativity, identity and inverses all hold, so H is a group (an abelian group, since the definition is symmetric in A and B). (A) is true and (B) is false.
6. Check (C): the complement $\overline{A} = X - A$ gives $A\Delta\overline{A} = A \cup \overline{A} = X$. This equals the identity $\varnothing$ only when $X = \varnothing$. For example, with $X = \{1\}$ and $A = \{1\}$, the complement is $\varnothing$ and $A\Delta\varnothing = A \neq \varnothing$. So (C) is false in general.
7. Correct options: (A) and (D).

KEY IDEA
Symmetric difference is addition mod 2 on indicator vectors: $(2^X, \Delta)$ is a group with identity $\varnothing$, and every element is its own inverse because $A\Delta A = \varnothing$.

WATCH OUT FOR
Confusing the inverse with the complement. $A\Delta\overline{A} = X$, not $\varnothing$, so the complement is not the inverse. Also, doubting associativity because the operation looks unusual.

Q.52 MSQ · 2 MARKS

Computer Networks › DNS & HTTP

Suppose in a web browser, you click on the www.gate-2023.in URL. The browser cache is empty. The IP address for this URL is not cached in your local host, so a DNS lookup is triggered (by the local DNS server deployed on your local host) over the 3-tier DNS hierarchy in an iterative mode. No resource records are cached anywhere across all DNS servers.

Let RTT denote the round trip time between your local host and DNS servers in the DNS hierarchy. The round trip time between the local host and the web server hosting www.gate-2023.in is also equal to RTT. The HTML file associated with the URL is small enough to have negligible transmission time and negligible rendering time by your web browser, which references 10 equally small objects on the same web server.

Which of the following statements is/are CORRECT about the minimum elapsed time between clicking on the URL and your browser fully rendering it?

(A) 7 RTTs, in case of non-persistent HTTP with 5 parallel TCP connections.

(B) 5 RTTs, in case of persistent HTTP with pipelining.

(C) 9 RTTs, in case of non-persistent HTTP with 5 parallel TCP connections.

(D) 6 RTTs, in case of persistent HTTP with pipelining.

Official answer: **C, D**

SOLUTION

1. The local DNS server runs on the local host, so the host-to-resolver hop costs nothing. In iterative mode it asks the root server, then the TLD (.in) server, then the authoritative server, one after another. Each query costs one RTT, so DNS takes $3\ \mathrm{RTT}$.

2. Fetching the base HTML file takes one RTT for the TCP handshake and one RTT for the HTTP request and response, so $2\ \mathrm{RTT}$.
3. Time before any referenced object can be requested: $3 + 2 = 5\ \mathrm{RTT}$.
4. Non-persistent HTTP with 5 parallel connections: every object needs a new TCP connection, which is one RTT for the handshake and one RTT for the request and response, so $2\ \mathrm{RTT}$ per object. Ten objects over 5 parallel connections need $\lceil 10/5 \rceil = 2$ rounds, which is $2 \times 2 = 4\ \mathrm{RTT}$.
5. Non-persistent total: $5 + 4 = 9\ \mathrm{RTT}$. So (C) is correct, and (A), which says 7 RTTs, is wrong.
6. Persistent HTTP with pipelining: the connection opened for the HTML file is reused, so no new handshake is needed. All 10 requests are sent back to back, and the responses return in $1\ \mathrm{RTT}$.
7. Persistent total: $5 + 1 = 6\ \mathrm{RTT}$. So (D) is correct, and (B), which says 5 RTTs, is wrong.

KEY IDEA

Add the stages in order. DNS costs 3 RTT (iterative, with the resolver on the host). The base page costs 2 RTT (handshake plus request). The 10 objects then cost 4 RTT in two rounds of 2 RTT each (non-persistent, 5 parallel), or 1 RTT in total (persistent with pipelining).

WATCH OUT FOR

Counting one RTT per object for the non-persistent case and forgetting the TCP handshake, which gives 7 instead of 9. Another slip is adding a fourth RTT for the host-to-resolver hop, or leaving out the handshake for the base page, which gives 5 for the persistent case.

Q.53 MSQ · 2 MARKS

Engineering Mathematics › Probability & statistics

Consider a random experiment where two fair coins are tossed. Let A be the event that denotes HEAD on both the throws, B be the event that denotes HEAD on the first throw, and C be the event that denotes HEAD on the second throw. Which of the following statements is/are TRUE?

(A) A and B are independent.

(B) A and C are independent.

(C) B and C are independent.

(D) $\mathrm{Prob}(B \mid C) = \mathrm{Prob}(B)$

Official answer: **C, D**

SOLUTION

1. The sample space of two fair coin tosses is $\{HH, HT, TH, TT\}$, each outcome with probability $\frac{1}{4}$.
2. The events are $A = \{HH\}$, $B = \{HH, HT\}$ and $C = \{HH, TH\}$. So $P(A) = \frac{1}{4}$, $P(B) = \frac{1}{2}$ and $P(C) = \frac{1}{2}$.
3. Events X and Y are independent exactly when $P(X \cap Y) = P(X)\,P(Y)$.
4. A and B: $A \cap B = \{HH\}$, so $P(A \cap B) = \frac{1}{4}$, while $P(A)P(B) = \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{8}$. These are not equal, so (A) is false.
5. A and C: $A \cap C = \{HH\}$, so $P(A \cap C) = \frac{1}{4} \neq \frac{1}{8} = P(A)P(C)$. So (B) is false.
6. B and C: $B \cap C = \{HH\}$, so $P(B \cap C) = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2} = P(B)P(C)$. So (C) is true.
7. $P(B \mid C) = \dfrac{P(B \cap C)}{P(C)} = \dfrac{1/4}{1/2} = \dfrac{1}{2} = P(B)$. So (D) is true. It restates the independence of B and C.

KEY IDEA

List the four equally likely outcomes and test independence with $P(X \cap Y) = P(X)P(Y)$. Two events are independent exactly when conditioning on one leaves the other's probability unchanged. So (C) and (D) say the same thing.

WATCH OUT FOR

Assuming that A (both heads) is independent of B or C because it involves the same tosses. In fact $A \subseteq B$ and $A \subseteq C$, so knowing B or C occurred raises the probability of A from $\frac{1}{4}$ to $\frac{1}{2}$.

Q.54 MSQ · 2 MARKS

Data Structures and Algorithms › Complexity

Consider functions Function_1 and Function_2 expressed in pseudocode as follows:

Function_1:

```
while n > 1 do
  for i = 1 to n do
     x = x + 1;
  end for
  n = ⌊n/2⌋;
end while
```

Function_2:

```
for i = 1 to 100*n do
  x = x + 1;
end for
```

Let $f_1(n)$ and $f_2(n)$ denote the number of times the statement "$x = x + 1$" is executed in Function_1 and Function_2, respectively.

Which of the following statements is/are TRUE?

(A) $f_1(n) \in \Theta(f_2(n))$

(B) $f_1(n) \in o(f_2(n))$

(C) $f_1(n) \in \omega(f_2(n))$

(D) $f_1(n) \in O(n)$

Official answer: **A, D**

SOLUTION

1. Function_1: the outer while loop runs while n>1, and each pass halves n via n=⌊n/2⌋. The inner for loop runs from i=1 to the *current* value of n, executing x=x+1 that many times per pass.
2. So f1(n) = n + ⌊n/2⌋ + ⌊n/4⌋ + … (continuing until n drops to 1), which, dropping floors (they don't affect asymptotic order), is a geometric series with ratio 1/2: n·(1 + 1/2 + 1/4 + …) which sums to at most 2n.
3. Hence f1(n) = Θ(n): it is bounded above by 2n and below by its leading term n (both linear in n), giving tight linear growth.
4. Function_2: the single for loop runs i=1 to 100·n, executing x=x+1 exactly 100n times, so f2(n) = 100n = Θ(n).
5. Option A (f1(n) ∈ Θ(f2(n))): since f1(n) ≈ 2n and f2(n) = 100n, the ratio f1(n)/f2(n) → 2/100, a positive constant, so both the upper- and lower-bound conditions for Θ hold. TRUE.
6. Option B (f1(n) ∈ o(f2(n))): little-o demands the ratio f1(n)/f2(n) → 0, but the ratio tends to the constant 1/50, not 0. FALSE.
7. Option C (f1(n) ∈ ω(f2(n))): little-omega demands the ratio → ∞, but again the ratio is a fixed constant. FALSE.
8. Option D (f1(n) ∈ O(n)): since f1(n) ≤ 2n for all sufficiently large n (constant c=2 suffices), this upper bound holds. TRUE.
9. Correct (TRUE) options: A and D.

KEY IDEA

Trace each loop's exact iteration count rather than guessing from loop nesting depth: `Function_1`'s outer `while` halves `n` each pass while the inner `for` runs the *current* `n`, giving a geometric series that sums to $\Theta(n)$, the same order as `Function_2`'s single pass of $100n$ iterations. Two functions of the same Θ order satisfy Θ (tight bound) and O (upper bound) with each other, but never o or ω (which require the ratio to actually vanish or blow up).

WATCH OUT FOR

Assuming `Function_1`'s nested while+for structure must be $\Theta(n \log n)$ or $\Theta(n^2)$ just because it looks like two loops, without summing the geometric series $n + n/2 + n/4 + \cdots$ (which totals $\Theta(n)$, not $\Theta(n \log n)$); a second common slip is treating "same Θ order" as automatically implying little-o or little-ω, when those need the ratio of the two functions to tend to 0 or ∞ respectively, not merely to be unequal.

Q.55 MSQ · 2 MARKS

Data Structures and Algorithms › Design techniques

Let G be a simple, finite, undirected graph with vertex set $\{v_1, \ldots, v_n\}$. Let $\Delta(G)$ denote the maximum degree of G and let $\mathbb{N} = \{1, 2, \ldots\}$ denote the set of all possible colors. Color the vertices of G using the following greedy strategy:

for $i = 1, \ldots, n$

$$\text{color}(v_i) \leftarrow \min\{j \in \mathbb{N} : \text{no neighbour of } v_i \text{ is colored } j\}$$

Which of the following statements is/are TRUE?

(A) This procedure results in a proper vertex coloring of G.

(B) The number of colors used is at most $\Delta(G) + 1$.

(C) The number of colors used is at most $\Delta(G)$.

(D) The number of colors used is equal to the chromatic number of G.

Official answer: **A, B**

SOLUTION

1. The algorithm processes vertices $v_1, \ldots, v_n$ in a fixed order, assigning each vertex the smallest color not already used by any of its already-colored neighbours (those earlier in the order).
2. (A) Properness: for any edge (v_i, v_k) with $i < k$, when v_k is colored, v_i is already colored, so v_k's rule explicitly excludes $\text{color}(v_i)$. Hence no edge is ever monochromatic, for any vertex order: the coloring produced is always proper. **(A) is TRUE.**
3. (B) Bound $\Delta(G) + 1$: when v_i is colored, its already-colored neighbours number at most $\Delta(G)$ (its total degree bound), so at most $\Delta(G)$ colors are forbidden. A color from $\{1, \ldots, \Delta(G) + 1\}$ is therefore always available, so greedy never uses more than $\Delta(G) + 1$ colors. **(B) is TRUE.**
4. (C) Tightness of the $\Delta(G)$ bound: take $G = K_n$, so $\Delta(G) = n - 1$. Every vertex is adjacent to all others, so each new vertex needs a brand-new color: colors $1, \ldots, n$ are all used, i.e. $n = \Delta(G) + 1 > \Delta(G)$ colors. So 'at most $\Delta(G)$' fails in general. **(C) is FALSE.**
5. (D) Order-dependence vs. chromatic number: greedy's color count depends on the chosen vertex ordering $v_1, \ldots, v_n$. A bipartite graph (e.g. a crown graph, $K_{n,n}$ minus a perfect matching) has $\chi(G) = 2$, but a poor ordering can force greedy to use strictly more than 2 colors. Since the number of colors used is not guaranteed to equal $\chi(G)$, **(D) is FALSE.**
6. Hence the TRUE statements are exactly (A) and (B).

KEY IDEA
Sequential greedy coloring always produces a proper coloring, and it never uses more than $\Delta(G)+1$ colors, but this bound is not always tight ($\Delta(G)$ alone can fail), and the actual number of colors greedy uses depends on the vertex order, so it need not equal $\chi(G)$.

WATCH OUT FOR
Confusing the *proved* greedy bound $\Delta(G)+1$ with a tighter, unproven bound of $\Delta(G)$, and assuming the greedy result equals the chromatic number $\chi(G)$ just because the algorithm is "optimal-looking"; the number of colors greedy uses is order-dependent and can exceed $\chi(G)$.

Q.56 NAT · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Let $U = \{1, 2, 3\}$. Let 2^U denote the powerset of U. Consider an undirected graph G whose vertex set is 2^U. For any $A, B \in 2^U$, (A, B) is an edge in G if and only if (i) $A \neq B$, and (ii) either $A \subsetneq B$ or $B \subsetneq A$. For any vertex A in G, the set of all possible orderings in which the vertices of G can be visited in a Breadth First Search (BFS) starting from A is denoted by $\mathcal{B}(A)$.

If $\emptyset$ denotes the empty set, then the cardinality of $\mathcal{B}(\emptyset)$ is _____.

Official answer: **5040**

SOLUTION

1. U = {1,2,3}, so 2^U has 2^3 = 8 elements; G has 8 vertices, one per subset of U.
2. Edge rule: A and B are adjacent in G iff one is a strict (proper) subset of the other: this is the FULL comparability relation of the subset partial order (any degree of nesting), not merely the Hasse-diagram cover edges.
3. ø is a proper subset of every one of the other 7 (nonempty) subsets of U, so ø is adjacent, directly, to all 7 remaining vertices in G.
4. Therefore a BFS starting at ø has exactly two levels: level 0 = {ø}, and level 1 = all 7 other vertices (each at distance 1); there is no level 2, since every vertex is already reachable from ø in one step.
5. BFS mechanics: after dequeuing ø, the algorithm scans ø's neighbor list and enqueues its unvisited neighbors. Since neither the graph nor the BFS specification fixes the order same-level neighbors are discovered, any of the 7! orders in which the 7 neighbors get enqueued is a valid BFS execution.
6. Once these 7 vertices are enqueued (in whichever order was chosen), dequeuing each in turn examines its own neighbors, but every vertex in G is already visited by that point, so no new vertex is discovered and the visiting order is not altered further; the final visiting order is exactly ø followed by the chosen permutation of the other 7.
7. Hence the number of distinct BFS visiting sequences starting from ø equals the number of permutations of the remaining 7 vertices: 7!.
8. 7! = 7×6×5×4×3×2×1 = 5040.

KEY IDEA
$\emptyset$ is a proper subset of every one of the other 7 subsets of U, so in G it is adjacent to all of them simultaneously. A BFS starting at $\emptyset$ therefore has only two levels: level 0 = $\{\emptyset\}$ and level 1 = all remaining 7 vertices. Since BFS's traversal order is not fixed for vertices within the same level (the specification leaves the order neighbors are discovered/enqueued unconstrained), every one of the 7! permutations of the 7 level-1 vertices is a valid BFS visiting order following $\emptyset$, and no further choices arise once they are enqueued (all vertices are already visited by then).

WATCH OUT FOR
Treating "adjacent iff proper subset" as only the Hasse-diagram cover relation (i.e., only edges to immediate subsets/supersets), which would make $\emptyset$ adjacent to just $\{1\}, \{2\}, \{3\}$ and force a multi-level BFS with far fewer orderings than 7!. The problem states the FULL comparability relation ($A \subsetneq B$ or $B \subsetneq A$, any degree of nesting), not just cover edges, so $\emptyset$ is directly adjacent to all 7 other vertices.

Q.57 NAT · 2 MARKS

Operating Systems › Virtual memory

Consider the following two-dimensional array D in the C programming language, which is stored in row-major order:

```
int D[128][128];
```

Demand paging is used for allocating memory and each physical page frame holds 512 elements of the array D. The Least Recently Used (LRU) page-replacement policy is used by the operating system. A total of 30 physical page frames are allocated to a process which executes the following code snippet:

```
for (int i = 0; i < 128; i++)
 for (int j = 0; j < 128; j++)
      D[j][i] *= 10;
```

The number of page faults generated during the execution of this code snippet is _____.

Official answer: **4096**

SOLUTION

1. Each page frame holds 512 elements and a row has 128 elements, so one page holds $512/128 = 4$ rows. The array occupies $128/4 = 32$ pages (taking `D` to start at a page boundary): page p holds rows $4p$ to $4p + 3$.
2. In `D[j][i]` the inner loop changes `j`, the row index. For a fixed `i` the references go down a column, through rows $0, 1, 2, \ldots, 127$.
3. Rows $4p$ to $4p + 3$ are on page p, so one pass over `j` references pages $0, 0, 0, 0, 1, 1, 1, 1, \ldots, 31, 31, 31, 31$. The repeated references inside a block of 4 rows are on a page that was just used, so they are hits.
4. Every pass over `j` therefore touches the 32 pages in the same cyclic order $0, 1, \ldots, 31$, and the next value of `i` starts again at page 0.
5. LRU with 30 frames on a cyclic reference of 32 distinct pages: the first 30 pages 0 to 29 fill the frames. Page 30 evicts page 0 and page 31 evicts page 1, so the frames hold $2, \ldots, 31$. The next pass asks for page 0, which is a fault and evicts the least recently used page, page 2. Page 1 is then also absent (fault, evicts page 3), and so on. The page that LRU evicts is always the one needed soonest, so every change of page is a fault.
6. Faults per value of `i` $= 32$, one for each page; the other 96 references in the pass are hits.
7. Total faults $= 128 \times 32 = 4096$.

KEY IDEA

The inner loop walks down a column of a row-major array, so it visits all 32 pages in a fixed cyclic order. LRU with fewer frames (30) than pages (32) on a cyclic reference faults on every page change.

WATCH OUT FOR

Assuming the inner loop reads row by row, which would give only 32 faults in total. Another slip is assuming that 30 of 32 frames means most pages stay resident, when LRU on a cycle evicts exactly the page needed next.

Q.58 NAT · 2 MARKS

Operating Systems › Memory management

Consider a computer system with 57-bit virtual addressing using multi-level tree-structured page tables with L levels for virtual to physical address translation. The page size is 4 KB (1 KB = 1024 B) and a page table entry at any of the levels occupies 8 bytes.

The value of L is _____.

Official answer: **5**

SOLUTION

1. The page size is $4\ \text{KB} = 2^{12}$ bytes, so the page offset takes 12 bits.
2. The virtual page number has $57 - 12 = 45$ bits.
3. Each page-table node is stored in one page. A page holds $\frac{2^{12}}{8} = 2^9 = 512$ entries, so one node holds 512 entries.
4. Each level therefore resolves $\log_2 512 = 9$ bits of the virtual page number.
5. The number of levels needed is $L = \left\lceil \frac{45}{9} \right\rceil = 5$. The 45 bits divide exactly into five groups of 9, so every level is completely full and no level can be dropped.
6. Check: with 4 levels only $4 \times 9 = 36 < 45$ bits are covered. With 5 levels $5 \times 9 = 45$ bits are covered exactly. So $L = 5$.

KEY IDEA
Each page-table node fits in one page, so each level resolves $\log_2(\text{page size}/\text{entry size})$ bits of the virtual page number. The number of levels is the ceiling of (virtual page number bits) divided by (bits per level).

WATCH OUT FOR
Dividing all 57 address bits by 9 instead of first removing the 12 offset bits. This gives $\lceil 57/9 \rceil = 7$, which is wrong. Another slip is using the entry size as 4 bytes, or forgetting that the top level must also fit in one page.

Q.59 NAT · 2 MARKS

Data Structures and Algorithms › Linear structures

Consider a sequence a of elements $a_0 = 1$, $a_1 = 5$, $a_2 = 7$, $a_3 = 8$, $a_4 = 9$, and $a_5 = 2$. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I: push the elements of a from a_0 to a_5 in that order into S.

II: enqueue the elements of a from a_0 to a_5 in that order into Q.

III: pop an element from S.

IV: dequeue an element from Q.

V: pop an element from S.

VI: dequeue an element from Q.

VII: dequeue an element from Q and push the same element into S.

VIII: Repeat operation VII three times.

IX: pop an element from S.

X: pop an element from S.

The top element of S after executing the above operations is ___ (answer as an integer).

Official answer: **8**

SOLUTION

1. Push $a_0..a_5$ into S in order: S (bottom→top) $= 1, 5, 7, 8, 9, 2$.
2. Enqueue $a_0..a_5$ into Q in order: Q (front→back) $= 1, 5, 7, 8, 9, 2$.
3. III: pop S removes 2; $S = 1, 5, 7, 8, 9$.
4. IV: dequeue Q removes 1; $Q = 5, 7, 8, 9, 2$.
5. V: pop S removes 9; $S = 1, 5, 7, 8$.

6. VI: dequeue Q removes 5; $Q = 7, 8, 9, 2$.
7. VII: dequeue Q removes 7, push 7 onto S; $Q = 8, 9, 2$; $S = 1, 5, 7, 8, 7$.
8. VIII (repeat VII three more times): dequeue 8 → push 8 ($Q = 9, 2$; $S = 1, 5, 7, 8, 7, 8$); dequeue 9 → push 9 ($Q = 2$; $S = 1, 5, 7, 8, 7, 8, 9$); dequeue 2 → push 2 ($Q = \text{empty}$; $S = 1, 5, 7, 8, 7, 8, 9, 2$).
9. IX: pop S removes 2; $S = 1, 5, 7, 8, 7, 8, 9$ (top $= 9$).
10. X: pop S removes 9; $S = 1, 5, 7, 8, 7, 8$ (top $= 8$).

KEY IDEA
Simulate the stack and queue step by step exactly in the order the operations are listed, tracking the full contents (not just the top/front) so each subsequent pop/dequeue references the correct element.

WATCH OUT FOR
Losing track of the moving front of Q while three separate dequeue-and-push actions happen in operation VIII (mixing up which element is currently at the front of Q), or forgetting that operation III pops before operation IV dequeues, so mis-ordering the interleaving of pops/dequeues against pushes/enqueues.

Q.60 NAT · 2 MARKS

Compiler Design › Syntax-directed translation

Consider the syntax directed translation given by the following grammar and semantic rules. Here N, I, F and B are non-terminals. N is the starting non-terminal, and #, 0 and 1 are lexical tokens corresponding to input letters "#", "0" and "1", respectively. X.val denotes the synthesized attribute (a numeric value) associated with a non-terminal X. I1 and F1 denote occurrences of I and F on the right hand side of a production, respectively. For the tokens 0 and 1, 0.val = 0 and 1.val = 1.

N → I # F N.val = I.val + F.val
I → I1 B I.val = (2 I1.val) + B.val
I → B I.val = B.val
F → B F1 F.val = (1/2)(B.val + F1.val)
F → B F.val = (1/2) B.val
B → 0 B.val = 0.val
B → 1 B.val = 1.val

The value computed by the translation scheme for the input string 10#011 is _____. (Rounded off to three decimal places)

Official answer: **2.374 to 2.376**

SOLUTION

1. Split the input $10\#011$ as $I\#F$ with $I \Rightarrow 10$ and $F \Rightarrow 011$. All attributes are synthesized, so evaluate bottom-up.
2. Integer part: $I_1 \Rightarrow 1$ uses $I \to B$, so $I_1.\text{val} = B.\text{val} = 1$. Then $I \to I_1 B$ with $B \Rightarrow 0$ gives $I.\text{val} = 2 \cdot 1 + 0 = 2$.
3. Fraction part, $F \Rightarrow 011$. The last digit comes from $F \to B$: $F.\text{val} = \frac{1}{2} \cdot 1 = 0.5$.
4. The second digit 1 uses $F \to BF_1$ with $F_1.\text{val} = 0.5$: $F.\text{val} = \frac{1}{2}(1 + 0.5) = 0.75$.
5. The first digit 0 uses $F \to BF_1$ with $F_1.\text{val} = 0.75$: $F.\text{val} = \frac{1}{2}(0 + 0.75) = 0.375$. This agrees with the binary fraction $0.011_2 = 0.375$.
6. At the root, $N \to I\#F$: $N.\text{val} = I.\text{val} + F.\text{val} = 2 + 0.375 = 2.375$.

KEY IDEA

The grammar reads the integer part as a binary number (each new digit doubles the value so far and adds itself), and the fraction part as a binary fraction built from the right: each digit adds to the value of the rest and halves it. So 10 gives 2, 011 gives 0.375, and the sum is 2.375.

WATCH OUT FOR

Evaluating the fraction from the left, or halving the digits in the wrong order. The rule $F \to BF_1$ puts the leftmost digit at weight 1/2, so the recursion must start from the last digit. Starting from the first digit gives a wrong value.

Q.61 NAT · 2 MARKS

Database Management and Warehousing › SQL

Q.61 Consider the following table named `Student` in a relational database. The primary key of this table is rollNum.

Student

rollNum	name	gender	marks
1	Naman	M	62
2	Aliya	F	70
3	Aliya	F	80
4	James	M	82
5	Swati	F	65

The SQL query below is executed on this database.

```
SELECT *
FROM Student
WHERE gender = 'F' AND
   marks > 65;
```

The number of rows returned by the query is _____.

Official answer: **2**

SOLUTION

1. The `WHERE` clause keeps a row only if both conditions hold: `gender = 'F'` and `marks > 65`.
2. Filter on gender first. Rows 2, 3 and 5 have gender F. Rows 1 and 4 are M, so they drop out.
3. Apply `marks > 65` to those three rows. Row 2 has 70, which passes. Row 3 has 80, which passes. Row 5 has 65, which fails because 65 is not strictly greater than 65.
4. Rows 2 and 3 satisfy both conditions. Their names are the same (Aliya), but `SELECT *` has no `DISTINCT` and the rows differ in rollNum, so both are returned.
5. The query returns 2 rows.

KEY IDEA

A `WHERE` with `AND` keeps only rows that satisfy both conditions. The comparison `marks > 65` is strict, so a row with exactly 65 is excluded. Without `DISTINCT`, rows are never merged, even if some columns repeat.

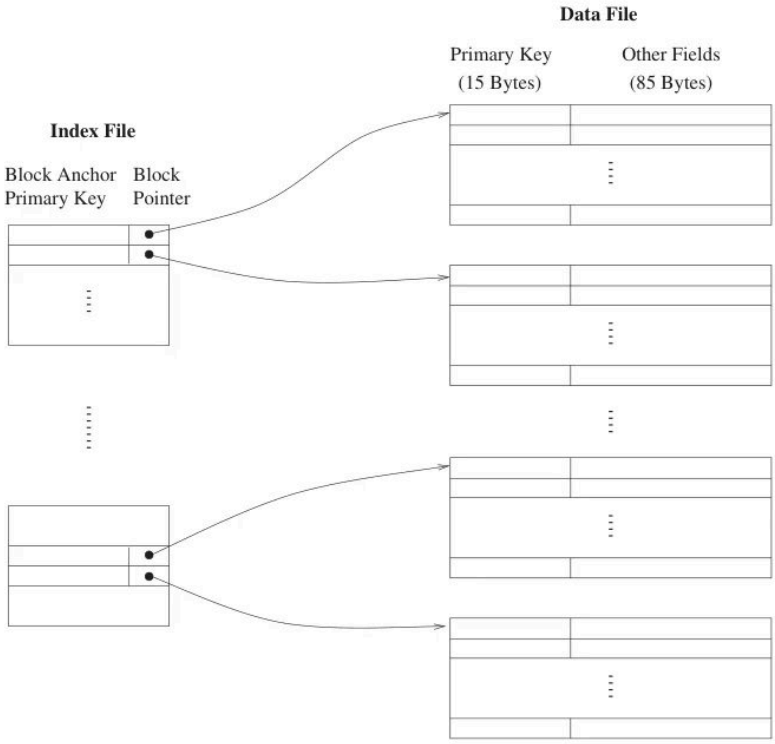
WATCH OUT FOR

Counting Swati (marks = 65) because 65 looks like it qualifies, which gives 3. Another slip is merging the two Aliya rows into one because the names match, which gives 1. Without `DISTINCT` both rows stay.

Q.62 NAT · 2 MARKS

Q.62

Consider a database of fixed-length records, stored as an ordered file. The database has 25,000 records, with each record being 100 bytes, of which the primary key occupies 15 bytes. The data file is block-aligned in that each data record is fully contained within a block. The database is indexed by a primary index file, which is also stored as a block-aligned ordered file. The figure below depicts this indexing scheme.



Suppose the block size of the file system is 1024 bytes, and a pointer to a block occupies 5 bytes. The system uses binary search on the index file to search for a record with a given key. You may assume that a binary search on an index file of b blocks takes $\lceil \log_2 b \rceil$ block accesses in the worst case.

Given a key, the number of block accesses required to identify the block in the data file that may contain a record with the key, in the worst case, is _____.

Official answer: **6**

SOLUTION

1. Records per data block $= \left\lfloor \frac{1024}{100} \right\rfloor = 10$. The file is block-aligned, so no record spans two blocks.
2. Data blocks $= \frac{25000}{10} = 2500$.
3. The primary index is sparse: one entry (block anchor primary key, block pointer) per data block, as the figure shows. So the index has 2500 entries.

4. Index entry size $= 15 + 5 = 20$ bytes. The index file is also block-aligned, so entries per index block $= \left\lfloor \frac{1024}{20} \right\rfloor = 51$.
5. Index blocks $b = \left\lceil \frac{2500}{51} \right\rceil = \lceil 49.02 \rceil = 50$.
6. Binary search on the index file takes $\lceil \log_2 50 \rceil = 6$ block accesses in the worst case, since $2^5 = 32 < 50 \leq 64 = 2^6$.
7. The question asks only for the accesses needed to IDENTIFY the data block that may hold the key. The index entry found by the binary search already carries that block's pointer, so no data block has to be read. The data-block read that would follow (to fetch the record) is not counted here.
8. Worst-case block accesses $= 6$.

KEY IDEA

A sparse primary index holds one (anchor key, block pointer) entry per data block. Count data blocks, then index blocks, then apply the given worst-case binary-search cost $\lceil \log_2 b \rceil$ to the index file alone.

WATCH OUT FOR

Adding 1 for reading the data block (giving 7). The question asks for the accesses needed to identify the block that may contain the record, not to fetch the record, so the data-block read is not counted. Other slips are building one index entry per record (a dense index, 25,000 entries) instead of one per data block, and using $1024/20 = 51.2$ entries per block without the floor.

Q.63 NAT · 2 MARKS

Theory of Computation › Finite automata

Consider the language L over the alphabet $\{0, 1\}$, given below:

$$L = \{w \in \{0,1\}^* \mid w \text{ does not contain three or more consecutive 1's}\}.$$

The minimum number of states in a Deterministic Finite-State Automaton (DFA) for L is _____.

Official answer: **4**

SOLUTION

1. A DFA for L only has to remember how many 1's have been read in a row (0, 1 or 2), and whether a run of three 1's has already occurred.
2. Let q_0 mean the current run of 1's has length 0 (the last symbol was 0, or nothing has been read yet). Let q_1 mean the run has length exactly 1, and q_2 mean it has length exactly 2. Let d mean three consecutive 1's have been seen. Once that happens, no later symbols can fix it, so d is a dead state.
3. Transitions: $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$; $\delta(q_1, 0) = q_0$, $\delta(q_1, 1) = q_2$; $\delta(q_2, 0) = q_0$, $\delta(q_2, 1) = d$; $\delta(d, 0) = \delta(d, 1) = d$. The start state is q_0.
4. Accepting states are q_0, q_1, q_2, because every string that reaches them has no run of three 1's. The dead state d is non-accepting.
5. Distinguishability check with suffixes. The empty string ε separates d from the other three states, since d rejects and the others accept. The suffix 1 separates q_2 from q_1 and from q_0: from q_2 it leads to d and is rejected, while from q_1 it leads to q_2 and from q_0 to q_1, both accepted. The suffix 11 separates q_1 from q_0: from q_1 it leads to d and is rejected, while from q_0 it leads to q_2 and is accepted.
6. All four states are pairwise distinguishable, so no two can be merged and this DFA is minimal. A complete DFA needs the dead state, so the minimum number of states is 4.

KEY IDEA

The language needs only a counter of the current run of 1's (0, 1, 2) plus one dead state for "three in a row seen". Minimality follows from showing every pair of states is distinguishable by a suffix.

WATCH OUT FOR

Leaving out the dead state and answering 3. A complete DFA must define a transition from every state on every symbol, so the trap state for "three consecutive 1's seen" counts as a fourth state.

Q.64 NAT · 2 MARKS

Computer Organization and Architecture › Memory & cache

An 8-way set associative cache of size 64 KB (1 KB = 1024 bytes) is used in a system with 32-bit address. The address is sub-divided into TAG, INDEX, and BLOCK OFFSET.

The number of bits in the TAG is _____.

Official answer: **19**

SOLUTION

1. Cache capacity = 64 KB = 65536 bytes; it is 8-way set associative; the address is 32 bits wide, split into TAG, INDEX and BLOCK OFFSET.
2. Let the (unstated) block size be B bytes. Number of blocks in the cache $= \dfrac{65536}{B}$. Since the cache is 8-way set associative, number of sets $= \dfrac{65536}{8B}$.
3. $\text{INDEX bits} = \log_2(\text{number of sets}) = \log_2\left(\dfrac{65536}{8B}\right)$, and $\text{OFFSET bits} = \log_2(B)$.
4. Adding them:
$$\text{INDEX} + \text{OFFSET} = \log_2\left(\frac{65536}{8B}\right) + \log_2(B) = \log_2\left(\frac{65536}{8}\right) = \log_2(8192) = 13 \text{ bits}$$
The block size B cancels out, so this total is the same for every possible block size, which is exactly why the question can be answered without ever specifying it.
5. $\text{TAG bits} = \text{Address width} - (\text{INDEX bits} + \text{OFFSET bits}) = 32 - 13 = 19$ bits.

KEY IDEA

In a set-associative cache, $\text{Cache size} = \text{Number of sets} \times \text{Ways} \times \text{Block size}$, so $\text{Number of sets} \times \text{Block size} = \dfrac{\text{Cache size}}{\text{Ways}}$, a quantity independent of block size. Since $\text{INDEX bits} + \text{OFFSET bits} = \log_2(\text{Number of sets} \times \text{Block size})$, this sum is fixed even though the block size itself is never stated in the question; only $\text{TAG bits} = \text{Address width} - (\text{INDEX} + \text{OFFSET})$ needs to be found, and it comes out the same for every possible block size.

WATCH OUT FOR

Assuming a block/line size must be given and getting stuck, or (wrongly) assuming a specific block size such as 32 or 64 bytes and computing a TAG value that depends on that unstated assumption; also forgetting to divide by the associativity (8-way) when computing the number of sets, which gives INDEX+OFFSET = 16 and TAG = 16 instead of 19.

Q.65 NAT · 2 MARKS

Computer Networks › IPv4

Q.65

The forwarding table of a router is shown below.

Subnet Number	Subnet Mask	Interface ID
200.150.0.0	255.255.0.0	1
200.150.64.0	255.255.224.0	2
200.150.68.0	255.255.255.0	3
200.150.68.64	255.255.255.224	4
Default		0

A packet addressed to a destination address 200.150.68.118 arrives at the router. It will be forwarded to the interface with ID _____.

Official answer: **3**

SOLUTION

1. A router forwards a packet by longest prefix match. For each entry, AND the destination 200.150.68.118 with the entry's mask and compare the result with the subnet number. Of all the entries that match, the one with the longest mask wins. The default route is used only when no other entry matches.
2. Row 1, mask 255.255.0.0 (/16). The first two octets are 200.150, so the result is 200.150.0.0. This equals the subnet number, so the entry matches with prefix length 16.
3. Row 2, mask 255.255.224.0 (/19). The third octet is $68 \,\&\, 224 = 01000100 \,\&\, 11100000 = 01000000 = 64$, and the fourth octet becomes 0. The result is 200.150.64.0, which equals the subnet number. The entry matches with prefix length 19.
4. Row 3, mask 255.255.255.0 (/24). The result is 200.150.68.0, which equals the subnet number. The entry matches with prefix length 24.
5. Row 4, mask 255.255.255.224 (/27). The first three octets are 200.150.68. The last octet is $118 \,\&\, 224 = 01110110 \,\&\, 11100000 = 01100000 = 96$. The result is 200.150.68.96, which is not 200.150.68.64. There is no match. (The block 200.150.68.64/27 covers 64 to 95, and 118 lies outside it.)
6. Rows 1, 2 and 3 match, with prefix lengths 16, 19 and 24. The longest is /24, which is row 3. The default route is not used because other entries match.
7. The packet is forwarded to interface 3.

KEY IDEA

Forwarding uses longest prefix match. Test every entry by ANDing the destination with its mask, keep the entries that match, and pick the one with the longest mask. Here that is 200.150.68.0/24, which gives interface 3.

WATCH OUT FOR

Choosing interface 4 because 200.150.68.64/27 has the longest mask and the address looks close to it. That entry covers 200.150.68.64 to 200.150.68.95 only, and 118 is outside this range, so it never matches. A second mistake is stopping at the first matching row (interface 1) or at the /19 row.

FROM IAE ACADEMY

Preparing for GATE CS?

IAE Academy teaches GATE CS in Ballygunge, Kolkata, and online. Counselling is free: ask us about batches, fees or your preparation.

FREE COUNSELLING

Chat on WhatsApp

Call +91 84204 36524

Open every day, 10:30 am to 7:30 pm.

Scan to chat on WhatsApp

THE GATE CS COURSE

Classroom batches in Ballygunge and online batches. Fees and details are on the course page.

Scan: course page

See the course

FREE ON IAEACADEMY.IN

More GATE CS papers, question by question with the official answers, and free notes by subject.

Scan: more papers

More solved papers

Found this useful? Send it to a friend preparing for GATE.

Send to a friend on WhatsApp

IAE Academy free study material. The questions are those of the GATE CS 2023 question paper, organised by IIT Kanpur; the answers are from its official answer key. Prepared and published by IAE Academy, Kolkata. Free to read and free to share, unaltered and with attribution. Not affiliated with, endorsed by or produced by IIT Kanpur or the GATE committee.

IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE coaching in Kolkata since 2012 · EC, EE, IN, CS and IT, DA, BT and CE · in the classroom and online

227A Rash Behari Ave, Ballygunge, Kolkata 700019

+91 84204 36524 · iaeacademy.in · Updated 27 October 2026

Get directions