IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE CS · 2020

IAE Academy free study material · Updated 27 October 2026

GATE CS 2020 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 Delhi

Online, with the questions linked to their topic notes: **iaeacademy.in/study/papers/cs-2020**

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: 42 MCQ · 23 NAT.

Official answer key: IIT Delhi

This paper numbers General Aptitude GA Q.1 to GA Q.10 and the subject questions Q.1 to Q.55: 65 questions in all.

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-2020

Answer key

Q	TYPE	MARKS	ANSWER
GA Q.1	MCQ	1	**D**
GA Q.2	MCQ	1	**A**
GA Q.3	MCQ	1	**A**
GA Q.4	MCQ	1	**D**
GA Q.5	MCQ	1	**B**
GA Q.6	MCQ	2	**D**
GA Q.7	MCQ	2	**C**
GA Q.8	MCQ	2	**A**
GA Q.9	MCQ	2	**B**
GA Q.10	MCQ	2	**C**
Q.1	MCQ	1	**A**
Q.2	MCQ	1	**A**
Q.3	MCQ	1	**C**
Q.4	MCQ	1	**C**
Q.5	MCQ	1	**B**
Q.6	MCQ	1	**C**
Q.7	MCQ	1	**Marks to all**
Q.8	MCQ	1	**D**
Q.9	MCQ	1	**D**
Q.10	MCQ	1	**C**
Q.11	MCQ	1	**C**
Q.12	MCQ	1	**C**
Q.13	MCQ	1	**A**
Q.14	MCQ	1	**A**
Q.15	MCQ	1	**D**
Q.16	MCQ	1	**C**
Q.17	NAT	1	**0.125**
Q.18	NAT	1	**7**
Q.19	NAT	1	**5**
Q.20	NAT	1	**1034**
Q.21	NAT	1	**13.3, or 13.5**
Q.22	NAT	1	**19**
Q.23	NAT	1	**13**
Q.24	NAT	1	**7**
Q.25	NAT	1	**6**
Q.26	MCQ	2	**A**
Q.27	MCQ	2	**C**
Q.28	MCQ	2	**B**
Q.29	MCQ	2	**B**
Q.30	MCQ	2	**B**
Q.31	MCQ	2	**D**
Q.32	MCQ	2	**A**
Q.33	MCQ	2	**B**
Q.34	MCQ	2	**A**
Q.35	MCQ	2	**B**
Q.36	MCQ	2	**A**
Q.37	MCQ	2	**A**
Q.38	MCQ	2	**B**
Q.39	MCQ	2	**C**
Q.40	MCQ	2	**A**
Q.41	MCQ	2	**B**
Q.42	NAT	2	**12**
Q.43	NAT	2	**2.15 to 2.18**
Q.44	NAT	2	**14**
Q.45	NAT	2	**0.5**
Q.46	NAT	2	**55**
Q.47	NAT	2	**511**
Q.48	NAT	2	**81**
Q.49	NAT	2	**99**
Q.50	NAT	2	**5.25**
Q.51	NAT	2	**6**
Q.52	NAT	2	**7**
Q.53	NAT	2	**154.5 to 155.5**
Q.54	NAT	2	**4**
Q.55	NAT	2	**44**

General Aptitude GA Q.1 to Q.10

GA Q.1 MCQ · 1 MARK

General Aptitude › Grammar & vocabulary

Raman is confident of speaking English _____ six months as he has been practising regularly _____ the last three weeks.

(A) during, for
(B) for, since
(C) for, in
(D) within, for

Official answer: **D**

SOLUTION

1. Blank 2: "the last three weeks" is a period of time that continues up to now, and the verb in that part of the sentence is in the present perfect continuous. "For" fits this ("for the last three weeks"). "Since" needs a starting point such as "since Monday", not a length of time, so B is wrong.
2. Blank 1: Raman is confident that he will be speaking English by the end of a period that starts now. "Within six months" means before six months have passed, so it fits this sense.
3. Option A: "during six months" means throughout a period, which does not express a deadline for a future skill. Option C: "for six months" says how long something lasts, so it cannot state when he will be able to speak English. Both are wrong at the first blank.
4. Check D: with "within" in the first blank and "for" in the second, the sentence reads "Raman is confident of speaking English within six months" and "... regularly for the last three weeks". Both blanks fit and the sentence is logical.

KEY IDEA
"Within" gives a time limit by which something will happen. "For" goes with a length of time that runs up to now. "Since" goes with a starting point.

WATCH OUT FOR
Choosing C because "for six months" sounds familiar. "For" tells how long something lasts. It does not say by when something will happen. Also, "in the last three weeks" is not wrong grammar, so the second blank cannot be used to reject C. The first blank rejects it.

GA Q.2 MCQ · 1 MARK

General Aptitude › Grammar & vocabulary

His knowledge of the subject was excellent but his classroom performance was _____.

(A) extremely poor
(B) good
(C) desirable
(D) praiseworthy

Official answer: **A**

SOLUTION

1. The conjunction "but" signals contrast, so the blank must oppose the first clause.
2. The first clause is positive: his knowledge of the subject was "excellent".
3. So the blank needs a negative description of his classroom performance.
4. Check the options: "good", "desirable" and "praiseworthy" are all positive. They would repeat the sense of the first clause, and "but" would no longer fit.
5. "extremely poor" is the only negative option, and it completes the contrast.

6. Hence the answer is (A).

KEY IDEA
A contrast word such as "but" requires the second clause to carry the opposite sense to the first. Here an excellent knowledge of the subject must be set against a negative description of performance.

WATCH OUT FOR
Picking a positive word such as "good" or "praiseworthy" without noticing that "but" demands a contrast.

GA Q.3 MCQ · 1 MARK

General Aptitude › Reasoning

Select the word that fits the analogy:

Cook : Cook :: Fly : _____

(A) Flyer

(B) Flying

(C) Flew

(D) Flighter

Official answer: **A**

SOLUTION

1. Read the analogy: in the first pair, "Cook" is the verb (the action) and the second "Cook" is the noun for the person who does it (a cook is one who cooks).
2. So the relation is: action (verb) : agent noun (the doer of the action).
3. Apply it to the verb "Fly": the doer of flying is a "Flyer" (one who flies, for example a frequent flyer).
4. Test the other options: "Flying" is a participle or gerund, not a doer; "Flew" is the past tense of the verb, so it keeps a verb relation rather than naming a doer; "Flighter" is not a standard English word.
5. Only "Flyer" gives verb : person who performs it, so the answer is (A).

KEY IDEA
Identify the relation in the given pair (verb : agent noun, the person who does the action) and apply the same relation to the second pair.

WATCH OUT FOR
Picking "Flew" or "Flying" because they are forms of the same verb. The first pair joins an action to its doer, not one verb form to another.

GA Q.4 MCQ · 1 MARK

General Aptitude › Reading

The dawn of the 21st century witnessed the melting glaciers oscillating between giving too much and too little to billions of people who depend on them for fresh water. The UN climate report estimates that without deep cuts to man-made emissions, at least 30% of the northern hemisphere's surface permafrost could melt by the end of the century. Given this situation of imminent global exodus of billions of people displaced by rising seas, nation-states need to rethink their carbon footprint for political concerns, if not for environmental ones.

Which one of the following statements can be inferred from the given passage?

(A) Nation-states do not have environmental concerns.

(B) Nation-states are responsible for providing fresh water to billions of people.

(C) Billions of people are responsible for man-made emissions.

(D) Billions of people are affected by melting glaciers.

Official answer: **D**

SOLUTION

1. Inference questions: pick the statement that follows directly from what the passage states, with no outside assumptions.
2. Option (A): the passage says nation-states need to rethink their carbon footprint for political concerns, 'if not for environmental ones'. This suggests environmental concerns are not the only reason to act. It does not say states have no environmental concerns, so (A) is not supported.
3. Option (B): the passage says billions of people depend on glaciers for fresh water. It never says nation-states are responsible for providing that water, so (B) is not supported.
4. Option (C): the passage links man-made emissions to melting permafrost. It never says the billions of people are responsible for those emissions, so (C) is not supported.
5. Option (D): the passage says billions of people depend on the melting glaciers for fresh water, and that billions face displacement from rising seas. So billions of people are affected by melting glaciers, and (D) follows directly from the passage.
6. Answer: (D).

KEY IDEA

In an inference question, choose only what the passage directly supports. Reject options that assign blame or responsibility, or that overstate a claim, unless the passage says so.

WATCH OUT FOR

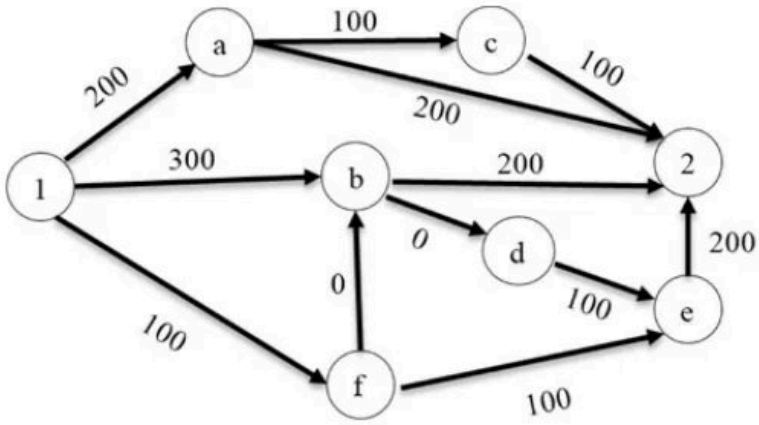
Reading 'if not for environmental ones' as 'states have no environmental concerns' (A). Or assuming that because states need to act, they are responsible for water supply (B), or that the people who depend on glaciers caused the emissions (C).

GA Q.5 MCQ · 1 MARK

General Aptitude › Numerical

Q.No. 5

There are multiple routes to reach from node 1 to node 2, as shown in the network.



The cost of travel on an edge between two nodes is given in rupees. Nodes 'a', 'b', 'c', 'd', 'e', and 'f' are toll booths. The toll price at toll booths marked 'a' and 'e' is Rs. 200, and is Rs. 100 for the other toll booths. Which is the cheapest route from node 1 to node 2?

(A) 1-a-c-2
(B) 1-f-b-2
(C) 1-b-2
(D) 1-f-e-2

Official answer: **B**

SOLUTION

1. Cost of a route = sum of the edge costs + the toll at every intermediate node passed. Tolls: a and e are Rs. 200; b, c, d, f are Rs. 100. Nodes 1 and 2 are the end points and carry no toll.
2. (A) 1-a-c-2: edges $200 + 100 + 100 = 400$; tolls at a and c $= 200 + 100 = 300$; total Rs. 700.
3. (B) 1-f-b-2: edges $100 + 0 + 200 = 300$; tolls at f and b $= 100 + 100 = 200$; total Rs. 500.
4. (C) 1-b-2: edges $300 + 200 = 500$; toll at b $= 100$; total Rs. 600.
5. (D) 1-f-e-2: edges $100 + 100 + 200 = 400$; tolls at f and e $= 100 + 200 = 300$; total Rs. 700.
6. The least total is Rs. 500, for route 1-f-b-2, which is option (B).

KEY IDEA

Total cost of a route = edge costs + tolls at the intermediate booths only. Work out each option in full and compare the totals.

WATCH OUT FOR

Leaving out the toll booths and comparing edge costs alone, or charging the toll differently at a and e. Edge costs alone give 300 for option B and 400 for options A and D, which hides the fact that tolls change the totals of C, A and D.

GA Q.6 MCQ · 2 MARKS

General Aptitude › Reading

Goods and Services Tax (GST) is an indirect tax introduced in India in 2017 that is imposed on the supply of goods and services, and it subsumes all indirect taxes except few. It is a destination-based tax imposed on goods and services used, and it is not imposed at the point of origin from where goods come. GST also has a few components specific to state governments, central government and Union Territories (UTs).

Which one of the following statements can be inferred from the given passage?

(A) GST is imposed on the production of goods and services.

(B) GST includes all indirect taxes.

(C) GST does not have a component specific to UT.

(D) GST is imposed at the point of usage of goods and services.

Official answer: **D**

SOLUTION

1. This is an inference question: the right option must follow from the passage itself, with no added assumption and no exaggeration.
2. Option (A): the passage says GST is imposed on the supply of goods and services, and not at the point of origin where goods come from. Production happens at the origin, so (A) goes against the passage.
3. Option (B): the passage says GST subsumes all indirect taxes except few. So some indirect taxes are not included, and (B) overstates the passage.
4. Option (C): the passage says GST has a few components specific to state governments, central government and Union Territories (UTs). So a UT-specific component does exist, and (C) is false.
5. Option (D): the passage says GST is a destination-based tax, imposed on goods and services used, and not at the point of origin. So it is imposed at the point of usage, and (D) follows directly.
6. Hence the answer is (D).

KEY IDEA
Pick the option the passage supports directly. Here "destination-based" and "imposed on goods and services used" mean the tax is charged where the goods and services are used.

WATCH OUT FOR
Choosing (B) because the passage says GST subsumes indirect taxes. The words "except few" mean not all are included. Choosing (C) by misreading "a few components specific to ... UTs" is another slip.

GA Q.7 MCQ · 2 MARKS

General Aptitude › Reasoning

If P = 3, R = 27, T = 243, then Q + S = _____.

(A) 40

(B) 80

(C) 90

(D) 110

Official answer: **C**

SOLUTION

1. Write the given values as powers of 3: $P = 3 = 3^1$, $R = 27 = 3^3$, $T = 243 = 3^5$.
2. P, R, T are the 1st, 3rd and 5th letters of P, Q, R, S, T, and their exponents 1, 3, 5 rise by 2 for every two letters. So the values form a geometric progression with the exponent rising by 1 per letter, and each letter equals 3^{position}.
3. Check the common ratio: $\frac{R}{P} = \frac{27}{3} = 9 = r^2$ and $\frac{T}{R} = \frac{243}{27} = 9 = r^2$, so $r = 3$. Both gaps agree, so the pattern fits the data.
4. Then $Q = P \times 3 = 9$ and $S = R \times 3 = 81$. The sequence is $3, 9, 27, 81, 243$.
5. $Q + S = 9 + 81 = 90$, which is option (C).

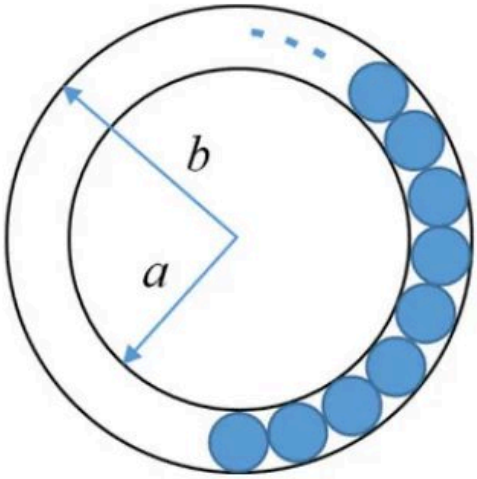
KEY IDEA
Rewrite the known terms as powers of one base. The letters P to T are consecutive, so P, R, T sit two places apart and the missing letters follow the same geometric pattern with ratio 3.

WATCH OUT FOR
Treating the values as an arithmetic progression (3, 27, 243 gaps) or adding ratios incorrectly. Another slip is taking the ratio between P and R as 9 per letter instead of 9 per two letters, which gives Q = 27 and a wrong sum.

GA Q.8 MCQ · 2 MARKS

General Aptitude › Geometry

Q.No. 8 The figure below shows an annular ring with outer and inner radii as b and a, respectively. The annular space has been painted in the form of blue colour circles touching the outer and inner periphery of annular space. If maximum n number of circles can be painted, then the unpainted area available in annular space is_____.



(A) $\pi\left[(b^2-a^2)-\frac{n}{4}(b-a)^2\right]$

(B) $\pi[(b^2-a^2)-n(b-a)^2]$

(C) $\pi\left[(b^2-a^2)+\frac{n}{4}(b-a)^2\right]$

(D) $\pi[(b^2-a^2)+n(b-a)^2]$

Official answer: **A**

SOLUTION

1. The annular space is the outer disc minus the inner disc, so its area is $\pi b^2 - \pi a^2 = \pi(b^2 - a^2)$.
2. Each painted circle touches both the inner and the outer periphery. So its diameter is the radial width of the ring: $d = b - a$.
3. Each painted circle therefore has radius $r = \dfrac{b-a}{2}$ and area $\pi r^2 = \dfrac{\pi(b-a)^2}{4}$.
4. The n painted circles together have area $\dfrac{n}{4}\pi(b-a)^2$.
5. Unpainted area = ring area minus painted area $= \pi\left[(b^2-a^2) - \dfrac{n}{4}(b-a)^2\right]$. This is option (A).

KEY IDEA
A circle that touches both boundaries of a ring has a diameter equal to the ring's width, $b - a$. Subtract the n circle areas from the ring area.

WATCH OUT FOR
Taking the painted circle's radius as $b - a$ instead of $(b - a)/2$. This gives $n(b - a)^2$ and option (B).

GA Q.9 MCQ · 2 MARKS

General Aptitude › Geometry

Two straight lines are drawn perpendicular to each other in X-Y plane. If α and β are the acute angles the straight lines make with the X-axis, then $\alpha + \beta$ is _____.

(A) 60°

(B) 90°

(C) 120°

(D) 180°

Official answer: **B**

SOLUTION

1. Let the first line make an inclination α with the X-axis, where $0 < \alpha < 90^\circ$, so it makes the acute angle α.
2. Two perpendicular lines differ in inclination by 90°, so the second line has inclination $\alpha + 90^\circ$, which is obtuse (between 90° and 180°).
3. The acute angle a line makes with the X-axis is the smaller of its inclination θ and $180^\circ - \theta$.
4. For the second line, $\beta = 180^\circ - (\alpha + 90^\circ) = 90^\circ - \alpha$.
5. Check with slopes: $\tan\alpha \cdot \tan(\alpha + 90^\circ) = -1$, and $\tan\beta = \tan(90^\circ - \alpha) = \cot\alpha$, so $\tan\alpha\tan\beta = 1$, consistent with $\alpha + \beta = 90^\circ$.
6. Therefore $\alpha + \beta = 90^\circ$, option (B).

KEY IDEA

Perpendicular lines have inclinations differing by 90°. The acute angle of the steeper-than-vertical (obtuse) line is its supplement, $180^\circ - (\alpha + 90^\circ) = 90^\circ - \alpha$, so the two acute angles are complementary.

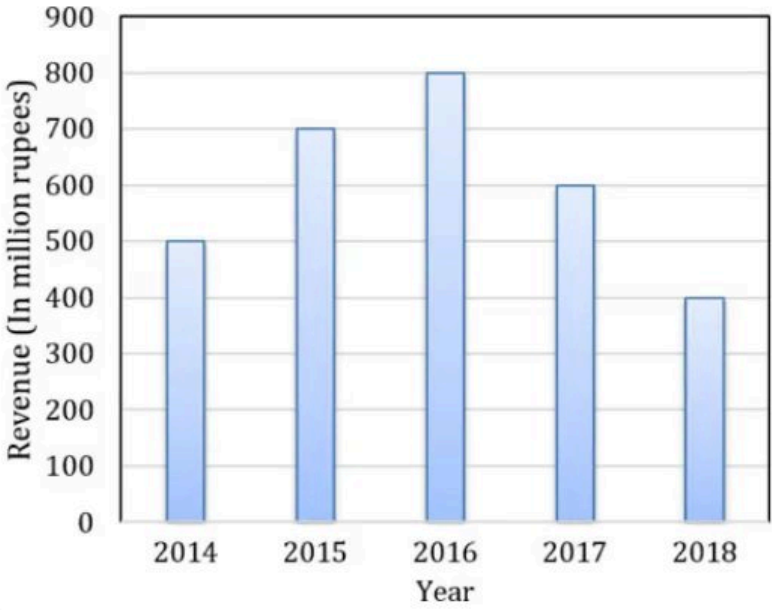
WATCH OUT FOR

Taking the second line's angle as $\alpha + 90^\circ$ and adding it directly (giving $2\alpha + 90^\circ$), or forgetting that the angle asked for is the acute one, which for an obtuse inclination is its supplement.

GA Q.10 MCQ · 2 MARKS

General Aptitude › Data interpretation

Q.No. 10 The total revenue of a company during 2014-2018 is shown in the bar graph. If the total expenditure of the company in each year is 500 million rupees, then the aggregate profit or loss (in percentage) on the total expenditure of the company during 2014-2018 is ____.



(A) 16.67 % profit
(B) 16.67 % loss
(C) 20 % profit
(D) 20 % loss

Official answer: **C**

SOLUTION

1. Read the bar heights from the graph: 2014 = 500, 2015 = 700, 2016 = 800, 2017 = 600, 2018 = 400 (million rupees).
2. Total revenue for 2014-2018: $500 + 700 + 800 + 600 + 400 = 3000$ million rupees.
3. Expenditure is 500 million rupees in each of the 5 years, so total expenditure $= 5 \times 500 = 2500$ million rupees.
4. Revenue is more than expenditure, so there is an aggregate profit: $3000 - 2500 = 500$ million rupees.
5. Profit percentage on total expenditure: $\frac{500}{2500} \times 100 = 20\%$.
6. The aggregate result is a 20 % profit, which is option (C).

KEY IDEA

Add the revenue of all five years and the expenditure of all five years first. Then take the aggregate profit or loss as a percentage of total expenditure, not of revenue.

WATCH OUT FOR

Dividing the profit by total revenue gives $500/3000 = 16.67\%$, which is option (A). The question asks for the percentage on expenditure. Another slip is averaging the yearly percentages instead of aggregating first.

Computer Science and Information Technology Q.1 to Q.55

Q.1 MCQ · 1 MARK

Engineering Mathematics › Calculus

Consider the functions

I. e^{-x}

II. $x^2 - \sin x$

III. $\sqrt{x^3 + 1}$

Which of the above functions is/are increasing everywhere in $[0, 1]$?

(A) III only

(B) II only

(C) II and III only

(D) I and III only

Official answer: **A**

SOLUTION

1. A differentiable function is increasing on an interval if $f'(x) \geq 0$ throughout it, with f' not zero on any whole sub-interval. A function with $f'(x) < 0$ anywhere in the interval is not increasing everywhere there.
2. I: $f(x) = e^{-x}$, so $f'(x) = -e^{-x} < 0$ for all x. It is decreasing, so I fails.
3. II: $f(x) = x^2 - \sin x$, so $f'(x) = 2x - \cos x$. At $x = 0$: $f'(0) = -1 < 0$. Since f' is continuous, f' stays negative on a small interval $[0, \delta)$, so f decreases there and II fails.
4. III: $f(x) = \sqrt{x^3 + 1}$, so $f'(x) = \dfrac{3x^2}{2\sqrt{x^3 + 1}}$. The denominator is positive on $[0, 1]$, so $f'(x) \geq 0$, and $f'(x) = 0$ only at $x = 0$. Hence f is increasing on all of $[0, 1]$.
5. Only III qualifies, so the answer is (A).

KEY IDEA

Test each function's derivative on the whole interval. Increasing everywhere means $f' \geq 0$ at every point. A single point with $f' < 0$ rules the function out.

WATCH OUT FOR

Judging II by the end of the interval. At $x = 1$, $f'(1) = 2 - \cos 1 > 0$, but $f'(0) = -1 < 0$, so II fails near 0.

Q.2 MCQ · 1 MARK

Data Structures and Algorithms › Complexity

For parameters a and b, both of which are $\omega(1)$, $T(n) = T(n^{1/a}) + 1$, and $T(b) = 1$.

Then $T(n)$ is

(A) $\Theta(\log_a \log_b n)$

(B) $\Theta(\log_{ab} n)$

(C) $\Theta(\log_b \log_a n)$

(D) $\Theta(\log_2 \log_2 n)$

Official answer: **A**

SOLUTION

1. Let $m = \log_b n$, so $n = b^m$. We track $S(m) := T(b^m)$.
2. Compute the effect of the recurrence's argument $n^{1/a}$ under this substitution: $n^{1/a} = (b^m)^{1/a} = b^{m/a}$, so $\log_b(n^{1/a}) = m/a$. Thus $T(n) = T(n^{1/a}) + 1$ becomes $S(m) = S(m/a) + 1$.
3. The base case $T(b) = 1$ corresponds to $m = \log_b b = 1$, i.e. $S(1) = 1$.

4. Unroll directly: $S(m) = S(m/a) + 1 = S(m/a^2) + 2 = \cdots = S(m/a^k) + k$. This reaches the base case $S(1)$ when $m/a^k = 1$, i.e. $k = \log_a m$. Hence $S(m) = \Theta(\log_a m)$ (no need to invoke the Master theorem, which applies to divide-and-conquer recurrences of the form $T(n) = aT(n/b) + f(n)$ and is not the right tool here; direct unrolling suffices).
5. Substitute back $m = \log_b n$: $T(n) = S(\log_b n) = \Theta(\log_a(\log_b n)) = \Theta(\log_a \log_b n)$, which is option (A).

KEY IDEA

Substituting $m = \log_b n$ turns the "take a-th root of the argument" recurrence $T(n) = T(n^{1/a}) + 1$ into the familiar "divide the exponent by a" recurrence $S(m) = S(m/a) + 1$, whose solution is $\Theta(\log_a m)$; undoing the substitution gives the nested double logarithm $\Theta(\log_a \log_b n)$.

WATCH OUT FOR

Treating the two parameters a and b as combinable into a single log base (picking option (B), $\Theta(\log_{ab} n)$) instead of recognizing that a controls how many times you must take roots (an outer log) while b sets the base at which the argument bottoms out (an inner log); the two play structurally different roles and cannot be merged into one log base.

Q.3 MCQ · 1 MARK

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

Consider the following statements.

I. Daisy chaining is used to assign priorities in attending interrupts.

II. When a device raises a vectored interrupt, the CPU does polling to identify the source of interrupt.

III. In polling, the CPU periodically checks the status bits to know if any device needs its attention.

IV. During DMA, both the CPU and DMA controller can be bus masters at the same time.

Which of the above statements is/are TRUE?

(A) I and II only
(B) I and IV only
(C) I and III only
(D) III only

Official answer: **C**

SOLUTION

1. Statement I: Daisy chaining connects devices along a single interrupt-acknowledge line; a device passes the acknowledge signal on only if it did not request interrupt, so the device closest to the CPU gets the highest priority. This is a standard hardware technique for assigning interrupt priority: TRUE.
2. Statement II: A vectored interrupt means the interrupting device itself supplies a vector (an address or device identifier) to the CPU during the interrupt-acknowledge cycle, so the CPU can directly locate the correct interrupt-service routine without checking each device in turn. Polling (checking status bits one by one) is the technique used for non-vectored interrupts, not vectored ones. So statement II is FALSE.
3. Statement III: This is the definition of polling: the CPU repeatedly reads the status/flag register of each device to see whether it needs service. This matches the statement exactly: TRUE.
4. Statement IV: During DMA, the DMA controller must request and be granted control of the system bus from the CPU (bus request / bus grant handshake); while the DMA controller holds the bus (in cycle-stealing, burst, or block mode), the CPU relinquishes bus mastership until the transfer or burst completes. The CPU and DMA controller are never simultaneous bus masters: FALSE.
5. True statements are I and III only, which is option (C).

KEY IDEA

Vectored interrupts let the device self-identify to the CPU (no polling needed); polling is only for non-vectored/software-poll schemes. Daisy chaining is a hardware priority-resolution mechanism. DMA requires exclusive bus mastership handed off from the CPU, never shared simultaneously.

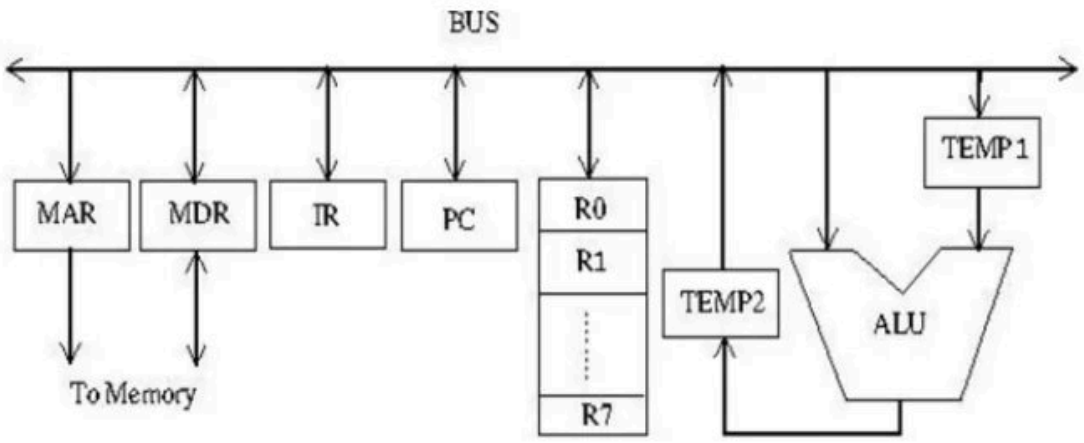
WATCH OUT FOR

Confusing "vectored" with requiring polling (students sometimes think the CPU still must poll to confirm the vector), and assuming DMA and CPU can share the bus concurrently since both are "active" during a DMA transfer. In reality bus mastership is exclusive at any instant.

Q.4 MCQ · 1 MARK

Computer Organization and Architecture › ALU & control unit

Q.No. 4 Consider the following data path diagram.



Consider an instruction: R0 ← R1 + R2. The following steps are used to execute it over the given data path. Assume that PC is incremented appropriately. The subscripts r and w indicate read and write operations, respectively.

1. $R2_r$, $TEMP1_r$, ALU_{add}, $TEMP2_w$
2. $R1_r$, $TEMP1_w$
3. PC_r, MAR_w, MEM_r
4. $TEMP2_r$, $R0_w$
5. MDR_r, IR_w

Which one of the following is the correct order of execution of the above steps?

(A) 2, 1, 4, 5, 3

(B) 1, 2, 4, 3, 5

(C) 3, 5, 2, 1, 4

(D) 3, 5, 1, 2, 4

Official answer: **C**

SOLUTION

1. Fetch phase: step 3 (PCr, MARw, MEMr) sends PC to MAR and starts a memory read; step 5 (MDRr, IRw) moves the fetched word from MDR into IR only once that read has completed. Hence step 3 must precede step 5, which already rules out options (A) and (B).
2. Execute phase for R0 ← R1 + R2: the ALU takes one operand directly off the bus and the other from the latched register TEMP1.

3. Step 2 (R1r, TEMP1w) places R1 on the bus and latches it into TEMP1.
4. Step 1 (R2r, TEMP1r, ALUadd, TEMP2w) places R2 on the bus while TEMP1 supplies its latched value to the ALU; ALUadd computes bus + TEMP1 and latches the sum into TEMP2. For this sum to be R1 + R2, TEMP1 must already hold R1, so step 2 must execute before step 1. This rules out option (D).
5. Step 4 (TEMP2r, R0w) drives the ALU result held in TEMP2 onto the bus and writes it into R0, so it must be the last step, after TEMP2 has been loaded by step 1.
6. Assembling: fetch (3, 5) then execute (2, 1, 4) gives 3, 5, 2, 1, 4: option (C).

KEY IDEA

In this single-bus datapath, the ALU has two inputs: one comes directly (combinationally) off the bus, the other from the latched register TEMP1. So whichever step places a value into TEMP1 must execute before the step that uses TEMP1 as an ALU input. Similarly, a memory read (MEMr) takes time to complete, so the step that issues it must finish before the step that consumes its result from MDR.

WATCH OUT FOR

Assuming the four steps that don't obviously belong to "fetch" (1, 2, 4) can be freely reordered as long as they come after the fetch pair, and picking option (D) by reading step 1 before step 2 in numeric order, without checking that step 1 needs TEMP1 to already hold R1, which only step 2 provides.

Q.5 MCQ · 1 MARK

Data Structures and Algorithms › Trees & heaps

The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19.
Which one of the following is the postorder traversal of the tree?

(A) 10, 11, 12, 15, 16, 18, 19, 20

(B) 11, 12, 10, 16, 19, 18, 20, 15

(C) 20, 19, 18, 16, 15, 12, 11, 10

(D) 19, 16, 18, 20, 11, 12, 10, 15

Official answer: **B**

SOLUTION

1. Root of the BST is the first preorder element: 15.
2. Split the remaining preorder elements into those less than 15 (left subtree preorder) and those greater than 15 (right subtree preorder): left = 10, 12, 11; right = 20, 18, 16, 19.
3. Reconstruct the left subtree from preorder (10, 12, 11): root 10; remaining values 12, 11 are both greater than 10, so they form 10's right subtree, whose preorder is 12, 11: root 12, with 11 (< 12) as its left child. So: 10 has right child 12, which has left child 11.
4. Reconstruct the right subtree from preorder (20, 18, 16, 19): root 20; remaining values 18, 16, 19 are all less than 20, so they form 20's left subtree, whose preorder is 18, 16, 19: root 18, then 16 (< 18) is its left child and 19 (> 18) is its right child. So: 20 has left child 18, which has left child 16 and right child 19.
5. Postorder visits (left subtree, then right subtree, then root) recursively. For the subtree rooted at 10: postorder(10) = postorder(right child 12) + [10]; postorder(12) = [11] (left child) + [12] = 11, 12. So postorder(10) = 11, 12, 10.
6. For the subtree rooted at 20: postorder(20) = postorder(left child 18) + [20]; postorder(18) = [16] (left) + [19] (right) + [18] = 16, 19, 18. So postorder(20) = 16, 19, 18, 20.
7. Combine at the overall root 15: postorder(whole tree) = postorder(left subtree, rooted 10) + postorder(right subtree, rooted 20) + [15] = 11, 12, 10, 16, 19, 18, 20, 15.
8. This sequence exactly matches option B.

KEY IDEA
A BST's preorder uniquely determines its shape: the root is first, and the remaining elements split into two contiguous runs, those less than the root (left subtree preorder) and those greater (right subtree preorder), recursively. Once the tree is reconstructed, postorder = left subtree postorder, then right subtree postorder, then root, applied recursively.

WATCH OUT FOR
Confusing preorder-to-postorder conversion with just reversing the preorder sequence (which would wrongly suggest something like 19,16,18,20,11,12,10,15, option D). Reversing preorder only produces a valid postorder-like listing for specific tree shapes, not in general; the visit order of left/right children must actually be respected via correct BST reconstruction.

Q.6 MCQ · 1 MARK

Data Structures and Algorithms › Trees & heaps

What is the worst case time complexity of inserting n² elements into an AVL-tree with n elements initially?

(A) $\Theta(n^4)$

(B) $\Theta(n^2)$

(C) $\Theta(n^2 \log n)$

(D) $\Theta(n^3)$

Official answer: **C**

SOLUTION

1. A single insertion into an AVL tree of m nodes costs $\Theta(\log m)$ in the worst case: $\Theta(\log m)$ to search down to the insertion point, plus $O(1)$ rotations to rebalance (each rotation is $O(1)$; checking balance factors back up the path is already counted in the $\Theta(\log m)$ search/traversal cost).
2. We start with n elements and insert n^2 more elements one at a time, so during this process the tree size m ranges from n (before the first new insertion) up to $n^2 + n$ (after the last).
3. For every value of m in this range, $\log m = \Theta(\log n)$: at the low end $\log n$ is trivially $\Theta(\log n)$, and at the high end $\log(n^2 + n) = 2\log n + \log\left(1 + \frac{1}{n}\right) = \Theta(\log n)$, since a constant multiple of $\log n$ is still $\Theta(\log n)$.
4. So every one of the n^2 insertions costs $\Theta(\log n)$ in the worst case, regardless of exactly where in the sequence it falls.
5. Total worst-case time $= n^2$ insertions $\times\ \Theta(\log n)$ per insertion $= \Theta(n^2 \log n)$.
6. This matches option (C): $\Theta(n^2 \log n)$.

KEY IDEA
Each AVL insertion costs Θ(log m) where m is the current tree size; since the tree size stays between n and n²+n throughout the process, log m is Θ(log n) for every single one of the n² insertions (because log(n²) = 2log n is still just a constant multiple of log n), so the n² insertions together cost Θ(n² log n), not Θ(n² log(n²)), because Θ absorbs the constant factor of 2.

WATCH OUT FOR
Writing the final tree size as n²+n and concluding the last insertions cost Θ(log(n²)) = Θ(2 log n²) and then sloppily treating the exponent as if it must appear explicitly, sometimes leading to Θ(n² log n²) being (wrongly) simplified to Θ(n⁴) or left as a distinct-looking but numerically identical expression to n² log n. Since Θ(log n²) = Θ(2 log n) = Θ(log n) exactly, both forms are the same complexity class, and no additional factor of n or n² should appear.

Q.7 MCQ · 1 MARK

Theory of Computation › Finite automata

Which one of the following regular expressions represents the set of all binary strings with an odd number of 1's?

(A) ((0 + 1)*1(0 + 1)*1)*10*

(B) (0∗10∗10∗)∗0∗1

(C) 10∗(0∗10∗10∗)∗

(D) (0∗10∗10∗)*10*

Official answer: **Marks to all**

Q.8 MCQ · 1 MARK

Theory of Computation › Regular languages

Consider the following statements.
I. If $L_1 \cup L_2$ is regular, then both L_1 and L_2 must be regular.
II. The class of regular languages is closed under infinite union.
Which of the above statements is/are TRUE?

(A) I only

(B) II only

(C) Both I and II

(D) Neither I nor II

Official answer: **D**

SOLUTION

1. Statement I: take $L_1 = \Sigma^*$, which is regular, and $L_2 = \{a^n b^n \mid n \geq 0\}$ over $\Sigma = \{a, b\}$, which is not regular (pumping lemma).
2. Then $L_1 \cup L_2 = \Sigma^*$ is regular, but L_2 is not regular. So Statement I is FALSE.
3. Statement II: closure of regular languages under union is guaranteed only for finitely many languages. It follows by induction from the two-language case, and induction does not extend to infinite unions.
4. Counterexample for an infinite union: each singleton $\{a^n b^n\}$ is finite, hence regular.
5. The union over all $n \geq 0$ is $\bigcup_{n \geq 0} \{a^n b^n\} = \{a^n b^n \mid n \geq 0\}$, which is not regular. So Statement II is FALSE.
6. Both statements are false, so the answer is (D) Neither I nor II.

KEY IDEA

Regular languages are closed under finite union only. Any language is a union of its singleton strings, and every singleton is regular, so closure under infinite union would make every language regular. Also, a union can be regular while a part of it is not, because Σ^* absorbs anything.

WATCH OUT FOR

Assuming that closure under union extends to infinite unions, or that a regular union forces both parts to be regular.

Q.9 MCQ · 1 MARK

Compiler Design › Lexical analysis

Consider the following statements.

I. Symbol table is accessed only during lexical analysis and syntax analysis.
II. Compilers for programming languages that support recursion necessarily need heap storage for memory allocation in the run-time environment.
III. Errors violating the condition 'any variable must be declared before its use' are detected during syntax analysis.

Which of the above statements is/are TRUE?

(A) I only

(B) I and III only

(C) II only

(D) None of I, II, and III

Official answer: **D**

SOLUTION

1. Statement I is false. The symbol table is used in more phases than lexical and syntax analysis. Semantic analysis looks names up in it for type checking and scope checking. Code generation and optimization read it for the type, size and storage of each name. So 'accessed only during lexical analysis and syntax analysis' is wrong.
2. Statement II is false. Recursion needs a fresh activation record for every call, and these records are allocated on the run-time stack in last-in, first-out order. Heap storage is needed only for data that outlives the call that created it, such as memory from `malloc` or `new`. A language with recursion but no dynamic allocation can run on a stack alone, so heap storage is not necessary.
3. Statement III is false. 'Every variable must be declared before its use' is a context-sensitive condition. A context-free grammar, which is what syntax analysis uses, cannot express it. The check is done in semantic analysis, by looking the identifier up in the symbol table.
4. All three statements are false, so the answer is (D) None of I, II, and III.

KEY IDEA

The symbol table is used in many phases, including semantic analysis and code generation. Recursion needs the run-time stack and not necessarily a heap. Declaration-before-use is a context-sensitive rule, so it is checked in semantic analysis and not by the context-free syntax analysis.

WATCH OUT FOR

Treating the symbol table as a tool only for the first two phases. Another slip is reading 'declared before use' as a syntax error, when it is a semantic one. A third is assuming recursion forces heap allocation, when stack frames are enough.

Q.10 MCQ · 1 MARK

Theory of Computation › Context-free languages

Consider the language $L = \{a^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$ and the following statements.

I. L is deterministic context-free.

II. L is context-free but not deterministic context-free.

III. L is not LL(k) for any k.

Which of the above statements is/are TRUE?

(A) I only

(B) II only

(C) I and III only

(D) III only

Official answer: **C**

Q.11 MCQ · 1 MARK

Operating Systems › Memory management

Consider allocation of memory to a new process. Assume that none of the existing holes in the memory will exactly fit the process's memory requirement. Hence, a new hole of smaller size will be created if allocation is made in any of the existing holes. Which one of the following statements is TRUE?

(A) The hole created by first fit is always larger than the hole created by next fit.

(B) The hole created by worst fit is always larger than the hole created by first fit.

(C) The hole created by best fit is never larger than the hole created by first fit.

(D) The hole created by next fit is never larger than the hole created by best fit.

Official answer: **C**

SOLUTION

1. The hole left after allocation is (size of the chosen hole) − (request size). The request size is the same for every strategy, so a strategy that picks a smaller hole leaves a smaller leftover hole. Only holes large enough for the request are candidates. By the question, no candidate fits exactly, so every candidate leaves a smaller hole.
2. Best fit picks the smallest adequate hole. First fit, next fit and worst fit each pick some adequate hole, so each picked hole is at least as large as the best-fit hole. Hence best fit leaves the smallest leftover of all four strategies.
3. (C) First fit picks an adequate hole, which is at least as large as the best-fit hole. So the hole created by best fit is never larger than the one created by first fit. TRUE.
4. (D) Next fit picks an adequate hole at least as large as the best-fit hole, so its leftover is never smaller than best fit's. It can be strictly larger: for example, if the next adequate hole after the pointer is bigger than the smallest adequate hole, next fit leaves a larger hole. So 'never larger than best fit' is FALSE.
5. (A) Take a case where the first adequate hole is also the first adequate hole found from the pointer. Then first fit and next fit pick the same hole and create equal holes, so 'always larger' fails. FALSE.
6. (B) Take a case where the first adequate hole is also the largest hole. Then first fit and worst fit pick the same hole and create equal holes, so 'always larger' fails. FALSE.
7. Only (C) is true.

KEY IDEA

Leftover hole = chosen hole size minus request size, so best fit, which picks the smallest adequate hole, always leaves the smallest leftover. Every other strategy leaves a hole at least as large as best fit's.

WATCH OUT FOR

Reading 'always larger' as true for worst fit or first fit. Two strategies can pick the very same hole, and then the leftovers are equal, so a strict 'always larger' claim fails. Another slip is assuming next fit can do no worse than best fit. It can leave a bigger hole.

Q.12 MCQ · 1 MARK

Operating Systems › Processes & threads

Consider the following statements about process state transitions for a system using preemptive scheduling.

I. A running process can move to ready state.
II. A ready process can move to running state.
III. A blocked process can move to running state.
IV. A blocked process can move to ready state.

Which of the above statements are TRUE?

(A) I, II, and III only

(B) II and III only

(C) I, II, and IV only

(D) I, II, III, and IV

Official answer: **C**

SOLUTION

1. Draw the five-state process model: New, Ready, Running, Blocked (Waiting), Terminated. The system uses preemptive scheduling, so a running process can be forced off the CPU.

2. Statement I: when a time slice expires or a higher-priority process arrives, the running process is preempted and goes back to the ready queue, Running → Ready. TRUE.
3. Statement II: the dispatcher picks a process from the ready queue and gives it the CPU, Ready → Running. TRUE.
4. Statement III: a blocked process is waiting for an event such as I/O completion. Even after the event occurs, it must first join the ready queue and wait for the dispatcher. There is no direct Blocked → Running edge. FALSE.
5. Statement IV: when the awaited I/O or event completes, the process moves Blocked → Ready. TRUE.
6. So I, II and IV are true and III is false. This is option (C).

KEY IDEA

A blocked process can only go to Ready when its event completes. Only the dispatcher moves a process from Ready to Running. Preemption adds the Running → Ready edge.

WATCH OUT FOR

Assuming a process whose I/O has finished resumes running at once (Blocked → Running). It must wait in the ready queue first, because the CPU may be held by another process.

Q.13 MCQ · 1 MARK

Database Management and Warehousing › SQL

Q.No. 13

Consider a relational database containing the following schemas.

Catalogue

sno	pno	cost
S1	P1	150
S1	P2	50
S1	P3	100
S2	P4	200
S2	P5	250
S3	P1	250
S3	P2	150
S3	P5	300
S3	P4	250

Suppliers

sno	sname	location
S1	M/s Royal furniture	Delhi
S2	M/s Balaji furniture	Bangalore
S3	M/s Premium furniture	Chennai

Parts

pno	pname	part_spec
P1	Table	Wood
P2	Chair	Wood
P3	Table	Steel
P4	Almirah	Steel
P5	Almirah	Wood

The primary key of each table is indicated by underlining the constituent fields.

```
SELECT s.sno, s.sname
FROM   Suppliers s, Catalogue c
WHERE  s.sno = c.sno AND
       cost > (SELECT AVG (cost)
               FROM Catalogue
               WHERE pno = 'P4'
               GROUP BY pno);
```

The number of rows returned by the above SQL query is

(A) 4
(B) 5
(C) 0
(D) 2

Official answer: **A**

SOLUTION

1. Run the subquery first. Filtering Catalogue to pno = 'P4' leaves $(S2, P4, 200)$ and $(S3, P4, 250)$.
2. GROUP BY pno forms a single group because every remaining row has pno = 'P4' . The subquery therefore returns one value, $\mathrm{AVG}(cost) = \frac{200+250}{2} = 225$. Since it is a scalar, the comparison cost > (subquery) is legal.
3. The outer query joins Suppliers s and Catalogue c on s.sno = c.sno . Every Catalogue row has a matching supplier, so the join keeps all 9 rows. Suppliers has no cost column, so the unqualified cost

means c.cost .

4. Keep the joined rows with $cost > 225$. For $S1$ the costs are 150, 50 and 100, so none qualify. For $S2$, $P4$ (200) fails and $P5$ (250) passes. For $S3$, $P1$ (250) passes, $P2$ (150) fails, $P5$ (300) passes and $P4$ (250) passes.
5. The qualifying rows are $(S2, P5)$, $(S3, P1)$, $(S3, P5)$ and $(S3, P4)$, which is 4 rows.
6. SELECT has no DISTINCT , so duplicate output rows stay. $(S3, \mathrm{M/s\ Premium\ furniture})$ appears 3 times. The query returns 4 rows, which is option (A).

KEY IDEA
Evaluate the scalar subquery first (the average cost of part P4 is 225). Then filter the joined rows with cost above 225 and count them. Without DISTINCT, repeated supplier rows are all counted.

WATCH OUT FOR
Counting distinct suppliers (2: S2 and S3) instead of joined rows, because `SELECT` without `DISTINCT` keeps duplicates. Another slip is using $>=$ instead of $>$, or averaging over all of `Catalogue` instead of only P4.

Q.14 MCQ · 1 MARK

Database Management and Warehousing › ER model

Which one of the following is used to represent the supporting many-one relationships of a weak entity set in an entity-relationship diagram?

(A) Diamonds with double/bold border
(B) Rectangles with double/bold border
(C) Ovals with double/bold border
(D) Ovals that contain underlined identifiers

Official answer: **A**

SOLUTION

1. A weak entity set has no key of its own. Its entities are identified only through an owner (identifying) entity set, so it must be linked to that owner by a supporting relationship.
2. The supporting relationship is many-one from the weak entity set to the owner entity set, and the weak entity set takes part in it totally.
3. In an ER diagram a weak entity set is drawn as a rectangle with a double (or bold) border. Its supporting (identifying) relationship is drawn as a diamond with a double (or bold) border.
4. The other options each show a different symbol. Option (B) is the weak entity set itself. Option (C) is a multivalued attribute. Option (D) is a key attribute, shown as an oval with an underlined identifier.
5. The supporting many-one relationship is therefore shown by option (A), diamonds with a double/bold border.

KEY IDEA
A weak entity set is a double-bordered rectangle. The relationship that ties it to its owner entity set is a double-bordered diamond. Do not mix up the shape of the entity set and the shape of its relationship.

WATCH OUT FOR
Choosing (B) because "double border" is associated with weak entities. The double-bordered rectangle is the weak entity set itself, not the supporting relationship. The question asks about the relationship, which is a diamond.

Q.15 MCQ · 1 MARK

Computer Networks › IPv4

Consider the following statements about the functionality of an IP based router.
I. A router does not modify the IP packets during forwarding.
II. It is not necessary for a router to implement any routing protocol.

III. A router should reassemble IP fragments if the MTU of the outgoing link is larger than the size of the incoming IP packet.
Which of the above statements is/are TRUE?

(A) I and II only
(B) I only
(C) II and III only
(D) II only

Official answer: **D**

SOLUTION

1. Statement I: when a router forwards an IP packet it decrements the TTL field, and because the header changed it must recompute the header checksum. The packet is therefore modified, so I is false.
2. Statement II: a router can forward packets using a statically configured forwarding table. No routing protocol such as RIP, OSPF or BGP is needed. So II is true.
3. Statement III: IP fragments are reassembled only at the destination host. A router does not reassemble them, even when the outgoing link has a larger MTU than the incoming packet. So III is false.
4. Only statement II is true, so the answer is (D) II only.

KEY IDEA
A router changes the TTL and header checksum on every hop, can run on static routes alone, and never reassembles fragments, because reassembly happens only at the destination host.

WATCH OUT FOR
Believing a router leaves packets untouched (forgetting the TTL decrement and checksum update), or that a router reassembles fragments when the next link has a larger MTU.

Q.16 MCQ · 1 MARK

Data Structures and Algorithms › Linear structures

What is the worst case time complexity of inserting n elements into an empty linked list, if the linked list needs to be maintained in sorted order?

(A) $\Theta(n)$
(B) $\Theta(n \log n)$
(C) $\Theta(n^2)$
(D) $\Theta(1)$

Official answer: **C**

SOLUTION

1. To insert a new element into a linked list that is already maintained in sorted order and currently holds k elements, the algorithm must start at the head and walk node by node, comparing the new value against each existing element, until it finds the correct position to splice the new node in.
2. In the worst case (for example, when the n elements arrive in strictly decreasing order, so each new element is smaller than every element already in the list and must be compared against all k existing elements before being inserted at the tail), this single insertion costs Θ(k) time. There is no way to avoid visiting each of the k nodes, because a linked list offers no random access to jump directly to a candidate position.
3. Inserting n elements one at a time, the i-th insertion (for i = 1, 2, …, n) is performed when the list already holds i − 1 elements, so its worst-case cost is Θ(i − 1). The total worst-case time over all n insertions is therefore Σ (i=1 to n) Θ(i−1) = Θ(0 + 1 + 2 + … + (n−1)).
4. The sum 0 + 1 + 2 + … + (n−1) equals n(n−1)/2 = (n² − n)/2, a standard triangular-number identity. As n grows, the n² term dominates and constant/lower-order terms are absorbed by asymptotic notation, so

this total cost is Θ(n²).

5. This quadratic bound cannot be improved: unlike a sorted array or a balanced BST, where the search for the correct position can exploit random access or tree structure to run in O(log n) time, a linked list only supports sequential traversal, so the position-search step for each insertion remains linear in the list's current size. Hence the overall worst-case time complexity of inserting n elements into an empty linked list while keeping it sorted is Θ(n²), which is option (C).

KEY IDEA
Maintaining sorted order in a linked list forces a linear scan from the head to find each element's correct position (no random access, so no binary search); summing the worst-case traversal cost 0+1+2+...+(n-1) over n insertions gives Θ(n²).

WATCH OUT FOR
Assuming a sorted linked list allows binary search (as with a sorted array), giving a false O(n log n) answer. Binary search requires random access, which a linked list does not have, so each insertion's search for position is inherently linear in the current list size.

Q.17 NAT · 1 MARK

Discrete Mathematics › Sets & relations

Let $\mathcal{R}$ be the set of all binary relations on the set $\{1, 2, 3\}$. Suppose a relation is chosen from $\mathcal{R}$ at random. The probability that the chosen relation is reflexive (round off to 3 decimal places) is _____.

Official answer: **0.125**

SOLUTION

1. A binary relation on $A = \{1, 2, 3\}$ is a subset of $A \times A$, and $|A \times A| = 3 \times 3 = 9$.
2. Each of the 9 ordered pairs is either in the relation or not, so the number of binary relations is $|\mathcal{R}| = 2^9 = 512$. Choosing at random makes each relation equally likely.
3. A relation is reflexive exactly when it contains $(1, 1)$, $(2, 2)$ and $(3, 3)$. These 3 pairs are forced.
4. The other $9 - 3 = 6$ pairs may each be present or absent, so the number of reflexive relations is $2^6 = 64$.
5. $$P = \frac{64}{512} = \frac{1}{8} = 0.125.$$

KEY IDEA
Count relations as subsets of $A \times A$. Reflexivity fixes the n diagonal pairs, leaving 2^{n^2-n} reflexive relations out of 2^{n^2}, so the probability is 2^{-n}. For $n = 3$ this is $1/8$.

WATCH OUT FOR
Counting only 2^3 relations, or forgetting that the 6 off-diagonal pairs are free. Another slip is treating the 3 diagonal pairs as optional, which gives the wrong ratio.

Q.18 NAT · 1 MARK

Discrete Mathematics › Groups

Let G be a group of 35 elements. Then the largest possible size of a subgroup of G other than G itself is _____.

Official answer: **7**

SOLUTION

1. By Lagrange's theorem, the order of every subgroup of a finite group divides the order of the group.
2. So a subgroup of G has order dividing $35 = 5 \times 7$, that is, one of $1, 5, 7, 35$.
3. Excluding G itself (order 35), the possible orders are $1, 5, 7$.
4. The largest of these is 7. It is attained: since 7 is a prime dividing $|G| = 35$, Cauchy's theorem gives an element of order 7. It generates a cyclic subgroup of order 7, which is a proper subgroup of G.
5. Hence the largest possible size of a proper subgroup of G is 7.

KEY IDEA
Lagrange's theorem restricts a subgroup's order to a divisor of 35. Cauchy's theorem shows the largest proper divisor, 7, is actually achieved.

WATCH OUT FOR
Stopping at Lagrange's theorem and taking the largest proper divisor without checking that a subgroup of that order exists. Divisibility is only necessary, and the existence of the order-7 subgroup comes from Cauchy's theorem.

Q.19 NAT · 1 MARK

Digital Electronics › Combinational

A multiplexer is placed between a group of 32 registers and an accumulator to regulate data movement such that at any given point in time the content of only one register will move to the accumulator. The minimum number of select lines needed for the multiplexer is _____.

Official answer: **5**

SOLUTION

1. The multiplexer must pass the content of exactly one of the 32 registers to the accumulator, so it is a 32-to-1 multiplexer with 32 data inputs.
2. A multiplexer with 2^k data inputs needs k select lines, because k select bits give 2^k distinct codes, one for each input.
3. Here $k = \log_2 32 = 5$, since $2^5 = 32$.
4. With 4 select lines only $2^4 = 16$ registers could be addressed, which is too few for 32. So the minimum number of select lines is 5.

KEY IDEA
A 2^k-to-1 multiplexer needs k select lines, so n inputs need $\lceil \log_2 n \rceil$ select lines.

WATCH OUT FOR
Answering 32 (the number of inputs) or 31 instead of $\log_2 32 = 5$. Select lines count address bits, not inputs.

Q.20 NAT · 1 MARK

Computer Organization and Architecture › Memory & cache

If there are m input lines and n output lines for a decoder that is used to uniquely address a byte addressable 1 KB RAM, then the minimum value of m + n is ____.

Official answer: **1034**

SOLUTION

1. A byte-addressable 1 KB RAM has $1\ \text{KB} = 2^{10} = 1024$ bytes, each requiring a unique address.
2. The decoder must provide exactly one output line per addressable byte, so $n = 1024$.
3. For a decoder, the number of input (address) lines m must satisfy $2^m \geq n$ so that every output line can be uniquely selected.
4. The minimum m occurs when $2^m = 1024$, i.e., $m = \log_2(1024) = 10$.
5. Minimum value of $m + n = 10 + 1024 = 1034$.

KEY IDEA
A decoder that uniquely addresses every byte of a 1 KB RAM needs one output line per byte and enough input (address) lines to select among all outputs of a full binary decoder ($n = 2^m$).

WATCH OUT FOR
Confusing "byte addressable 1 KB" with the number of address bits alone (giving m + m = 20) or forgetting that n must equal the full 1024 output lines (one per addressable byte), not some smaller count.

Q.21 NAT · 1 MARK

Computer Organization and Architecture › Memory & cache

A direct mapped cache memory of 1 MB has a block size of 256 bytes. The cache has an access time of 3 ns and a hit rate of 94%. During a cache miss, it takes 20 ns to bring the first word of a block from the main memory, while each subsequent word takes 5 ns. The word size is 64 bits. The average memory access time in ns (round off to 1 decimal place) is ____.

Official answer: **13.3, or 13.5**

SOLUTION

1. Word size = 64 bits = 8 bytes. Words per block = block size / word size = 256 bytes / 8 bytes = 32 words.
2. On a miss, the first word costs 20 ns and each of the remaining 31 words costs 5 ns, so miss penalty = 20 + 31 \times 5 = 20 + 155 = 175 \text{ ns}.
3. Miss rate = 1 - hit rate = 1 - 0.94 = 0.06.
4. AMAT = hit time + miss rate \times miss penalty = 3 + 0.06 \times 175 = 3 + 10.5 = 13.5 \text{ ns}.

KEY IDEA

Average memory access time (AMAT) combines the cache's own access time with the extra time paid only on a miss: $\text{AMAT} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$, where the miss penalty for a block transfer is (time for first word) + (words remaining) $\times$ (time per subsequent word).

WATCH OUT FOR

Using the wrong word size to compute words per block (e.g. forgetting to convert 64 bits to 8 bytes), or forgetting to subtract 1 from the word count when computing the miss penalty (the first word already costs 20 ns, so only the remaining 31 words cost 5 ns each). This pair of slips is exactly why the answer key accepts two nearby values.

Q.22 NAT · 1 MARK

Programming in C › Pointers & arrays

Consider the following C program.

```
#include <stdio.h>
int main() {
   int a[4][5]={{1, 2, 3, 4, 5},
                {6, 7, 8, 9, 10},
                {11, 12, 13, 14, 15},
                {16, 17, 18 , 19, 20}};
   printf("%d\n", *(*(a+**a+2)+3));
   return (0);
}
```

The output of the program is _____.

Official answer: **19**

SOLUTION

1. a is declared as int a[4][5] = {{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15},{16,17,18,19,20}}.
2. *a is a[0], the first row, which decays to a pointer of type int* pointing at a[0][0].
3. **a dereferences that pointer, giving a[0][0] = 1.
4. a + **a + 2 = a + 1 + 2 = a + 3. Since a has type int (*)[5] (pointer to a row of 5 ints), a+3 points to the whole row a[3], not to an individual int.
5. *(a+3) is a[3], which decays to an int* pointing at a[3][0] (value 16).
6. *(a+3) + 3 performs ordinary pointer arithmetic on this int*, advancing 3 ints from a[3][0] to a[3][3].
7. *(*(a+3)+3) dereferences that pointer, giving a[3][3].

8. Row a[3] = {16, 17, 18, 19, 20} ; index 3 (0-based) is 19.
9. printf("%d\n", ...) prints this int, so the program's output is 19.

KEY IDEA

For a 2D array `int a[4][5]` , `a` decays to a pointer of type `int(*)[5]` (pointer to a row), so `a + k` skips `k` whole rows. Inside a row, ordinary pointer arithmetic on an `int*` skips single ints. Evaluate the expression from the innermost `*` outward, tracking the type at each step.

WATCH OUT FOR

Confusing `**a` (which decays through two levels to give `a[0][0] = 1` , an int) with `*a` (which is `a[0]` , decaying to a pointer to `int`), and then applying the wrong pointer arithmetic scale (rows of size 5 vs. single ints) at each step of `*(*(a+**a+2)+3)` .

Q.23 NAT · 1 MARK

Data Structures and Algorithms › Sorting & hashing

Consider a double hashing scheme in which the primary hash function is $h_1(k) = k \bmod 23$, and the secondary hash function is $h_2(k) = 1 + (k \bmod 19)$. Assume that the table size is 23. Then the address returned by probe 1 in the probe sequence (assume that the probe sequence begins at probe 0) for key value $k = 90$ is ___.

Official answer: **13**

SOLUTION

1. Double hashing probe formula: $\text{address}(i) = (h_1(k) + i \cdot h_2(k)) \bmod 23$, with probe 0 corresponding to $i = 0$.
2. Compute $h_1(90) = 90 \bmod 23$. Since $23 \times 3 = 69$, the remainder is $90 - 69 = 21$, so $h_1(90) = 21$.
3. Compute $h_2(90) = 1 + (90 \bmod 19)$. Since $19 \times 4 = 76$, the remainder is $90 - 76 = 14$, so $90 \bmod 19 = 14$, giving $h_2(90) = 1 + 14 = 15$.
4. Probe 0 address = $(21 + 0 \times 15) \bmod 23 = 21$.
5. Probe 1 address = $(21 + 1 \times 15) \bmod 23 = 36 \bmod 23 = 13$.
6. So the address returned by probe 1 is 13.

KEY IDEA

Double hashing: the probe sequence address at step i is $\text{address}(i) = (h_1(k) + i \cdot h_2(k)) \bmod m$, where probe 0 is the first probe ($i = 0$).

WATCH OUT FOR

Numbering probe 0 as $i = 1$ (an off-by-one), which would compute probe 1 as $i = 2$ and give the wrong address; or computing $h_2(k)$ without the $+1$ offset.

Q.24 NAT · 1 MARK

Compiler Design › Parsing

Consider the following grammar.

$$S \to aSB \mid d$$

$$B \to b$$

The number of reduction steps taken by a bottom-up parser while accepting the string `aaadbbb` is ___.

Official answer: **7**

SOLUTION

1. A bottom-up parser traces a rightmost derivation in reverse. Each reduction undoes one production use, so the number of reductions equals the number of steps in the rightmost derivation.
2. Rightmost derivation of aaadbbb :

$$S \Rightarrow aSB \Rightarrow aSb \Rightarrow aaSBb \Rightarrow aaSbb \Rightarrow aaaSBbb \Rightarrow aaaSbbb \Rightarrow aaadbbb$$

Each step rewrites the rightmost nonterminal, and the final string is aaadbbb .

3. Count the productions used: $S \to aSB$ three times (one for each leading a), $B \to b$ three times (one for each trailing b), and $S \to d$ once.
4. Parser run: shift aaad, then reduce $S \leftarrow d$ (1 reduction). The stack is aaaS. Shift b, reduce $B \leftarrow b$, then reduce $S \leftarrow aSB$. Repeat this shift and two reductions for each of the three b s. The stack ends as S with the input empty, so the string is accepted.
5. Total reductions: $1 + 3 \times 2 = 7$.

KEY IDEA
The number of reductions equals the number of production applications in the rightmost derivation. Here that is 3 uses of S → aSB, 1 use of S → d and 3 uses of B → b.

WATCH OUT FOR
Counting only the reductions by S → aSB and S → d, which gives 4, and forgetting the three B → b reductions. Another slip is counting shift moves as steps.

Q.25 NAT · 1 MARK

Computer Networks › DNS & HTTP

Assume that you have made a request for a web page through your web browser to a web server. Initially the browser cache is empty. Further, the browser is configured to send HTTP requests in non-persistent mode. The web page contains text and five very small images. The minimum number of TCP connections required to display the web page completely in your browser is _____.

Official answer: **6**

SOLUTION

1. With non-persistent HTTP, every object is fetched over its own TCP connection, and the server closes that connection after sending the object.
2. The browser cache is empty, so nothing can be reused. The browser has to fetch the base HTML file, which holds the text, and then each of the five images.
3. The base HTML file needs 1 TCP connection.
4. Each of the 5 images is a separate object and needs its own connection: $5 \times 1 = 5$.
5. Total connections $= 1 + 5 = 6$. Each object is fetched exactly once, so this is the minimum.

KEY IDEA
In non-persistent HTTP, each object (the base HTML file and every embedded image) needs its own TCP connection. With an empty cache, the number of connections is 1 + the number of images.

WATCH OUT FOR
Counting only 5 connections (the images alone) and forgetting the base HTML page. Another slip is to assume one connection can be reused for all objects, which is persistent HTTP and not what is asked here.

Q.26 MCQ · 2 MARKS

Theory of Computation › Turing machines

Which of the following languages are undecidable? Note that $\langle M \rangle$ indicates encoding of the Turing machine M.

$L_1 = \{\langle M \rangle \mid L(M) = \emptyset\}$

$L_2 = \{\langle M, w, q \rangle \mid M \text{ on input } w \text{ reaches state } q \text{ in exactly } 100 \text{ steps}\}$

$L_3 = \{\langle M \rangle \mid L(M) \text{ is not recursive}\}$

$L_4 = \{\langle M \rangle \mid L(M) \text{ contains at least } 21 \text{ members}\}$

(A) L_1, L_3, and L_4 only

(B) L_1 and L_3 only

(C) L_2 and L_3 only

(D) L_2, L_3, and L_4 only

Official answer: **A**

SOLUTION

1. L_2: whether M on input w reaches state q in exactly 100 steps needs only a bounded simulation. Run M on w for 100 steps and see whether it is in state q after step 100. This always halts, so L_2 is decidable.
2. $L_1 = \{\langle M\rangle \mid L(M) = \emptyset\}$ asks about the language $L(M)$ alone. The property is non-trivial: a machine that rejects everything has $L(M) = \emptyset$, and a machine that accepts Σ^* does not. By Rice's theorem L_1 is undecidable.
3. $L_3 = \{\langle M\rangle \mid L(M) \text{ is not recursive}\}$ also asks about $L(M)$ alone. It is non-trivial: $L(M) = \emptyset$ is recursive, so a machine that accepts nothing is not in L_3. The halting language H is recursively enumerable but not recursive, so a machine accepting H is in L_3. By Rice's theorem L_3 is undecidable.
4. $L_4 = \{\langle M\rangle \mid L(M) \text{ contains at least 21 members}\}$ is a property of $L(M)$ alone. It is non-trivial: $\emptyset$ fails it and Σ^* satisfies it. By Rice's theorem L_4 is undecidable.
5. So L_1, L_3 and L_4 are undecidable and L_2 is decidable. The answer is option (A), L_1, L_3, and L_4 only.

KEY IDEA
Rice's theorem makes every non-trivial property of $L(M)$ undecidable. A property of a bounded run of M, such as exactly 100 steps, is decidable by direct simulation.

WATCH OUT FOR
Treating L_2 as undecidable because it mentions a Turing machine's run. The step bound makes it a finite simulation that always halts. Another slip is to skip L_3 because it says "not recursive", although it is still a non-trivial property of the language.

Q.27 MCQ · 2 MARKS

Engineering Mathematics › Linear algebra

Let A and B be two $n \times n$ matrices over real numbers. Let $\text{rank}(M)$ and $\det(M)$ denote the rank and determinant of a matrix M, respectively. Consider the following statements.

I. $\text{rank}(AB) = \text{rank}(A)\,\text{rank}(B)$

II. $\det(AB) = \det(A)\det(B)$

III. $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$

IV. $\det(A + B) \leq \det(A) + \det(B)$

Which of the above statements are TRUE?

(A) I and II only

(B) I and IV only

(C) II and III only

(D) III and IV only

Official answer: **C**

SOLUTION

1. **I.** Take $A = B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. Then $\text{rank}(A)\,\text{rank}(B) = 1 \cdot 1 = 1$, but $AB = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ has rank 0. So I is false. In general only $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$ holds.
2. **II.** The determinant is multiplicative for square matrices of the same size: $\det(AB) = \det(A)\det(B)$. II is true.
3. **III.** Every column of $A + B$ is a sum of a column of A and a column of B, so the column space of $A + B$ lies inside (column space of A) + (column space of B). That sum has dimension at most $\text{rank}(A) +$

$\text{rank}(B)$, so $\text{rank}(A+B) \le \text{rank}(A) + \text{rank}(B)$. III is true.

4. **IV.** Take $A = B = I_2$. Then $\det(A+B) = \det(2I_2) = 4$, while $\det(A) + \det(B) = 1 + 1 = 2$. Since $4 \le 2$ is false, IV is false.
5. Only II and III are true, which is option (C).

KEY IDEA

The determinant is multiplicative, and rank is subadditive under addition. Rank is not multiplicative under products, and the determinant is not subadditive under addition.

WATCH OUT FOR

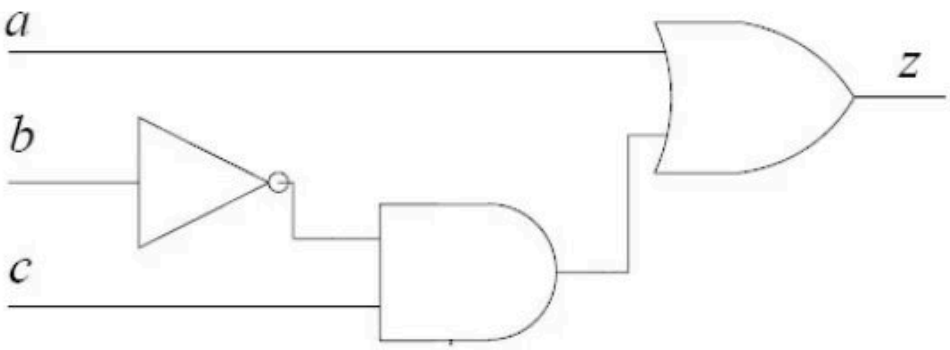
Treating rank and determinant alike. Rank of a product is only bounded by the smaller rank, not equal to the product of ranks. Determinant of a sum is not bounded by the sum of determinants. Both are refuted by simple counterexamples (a nilpotent matrix, and the identity matrix).

Q.28 MCQ · 2 MARKS

Digital Electronics › Boolean logic

Q.No. 28

Consider the Boolean function $z(a, b, c)$.



Which one of the following minterm lists represents the circuit given above?

(A) $z = \sum(0, 1, 3, 7)$

(B) $z = \sum(1, 4, 5, 6, 7)$

(C) $z = \sum(2, 4, 5, 6, 7)$

(D) $z = \sum(2, 3, 5)$

Official answer: **B**

SOLUTION

1. Trace the gates: b passes through a NOT gate, giving $\overline{b}$. The AND gate receives $\overline{b}$ and c, so its output is $\overline{b}c$.
2. The OR gate combines a with that output: $z = a + \overline{b}c$.
3. Number the minterms by $4a + 2b + c$. The term a is 1 whenever $a = 1$, which covers $100, 101, 110, 111$, that is minterms $4, 5, 6, 7$.
4. The term $\overline{b}c$ is 1 when $b = 0$ and $c = 1$. For $a = 0$ this is $001 = 1$. For $a = 1$ it is $101 = 5$, which is already counted.

5. Union: $z = \sum m(1, 4, 5, 6, 7)$.
6. Check with the truth table:

a	b	c	$\bar{b}c$	z
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

   The function is 1 exactly at minterms $1, 4, 5, 6, 7$, and 0 at $0, 2, 3$.
7. Compare with the options. Only (B) lists $1, 4, 5, 6, 7$. (A) wrongly includes 0 and 3. (C) wrongly includes 2 and omits 1. (D) is far too short.

KEY IDEA

Write the output of the circuit as a Boolean expression, $z = a + \bar{b}c$. Then list the input combinations that make it 1 using the minterm index $4a + 2b + c$, with a as the most significant bit.

WATCH OUT FOR

Reading the bits in the wrong order, for example with c as the most significant bit, which gives a different minterm list. Another slip is forgetting that a alone makes $z = 1$ for all four combinations of b and c. A third is dropping the bubble on b and using bc in place of $\bar{b}c$.

Q.29 MCQ · 2 MARKS

Computer Organization and Architecture › ALU & control unit

Consider three registers R1, R2, and R3 that store numbers in IEEE-754 single precision floating point format. Assume that R1 and R2 contain the values (in hexadecimal notation) 0x42200000 and 0xC1200000, respectively.

If $R3 = \dfrac{R1}{R2}$, what is the value stored in R3?

(A) 0x40800000

(B) 0xC0800000

(C) 0x83400000

(D) 0xC8500000

Official answer: **B**

SOLUTION

1. Decode R1 = 0x42200000: binary 0100 0010 0010 0000...0000. Sign = 0 (positive). Exponent bits = 10000100 = 132; unbiased exponent = 132 - 127 = 5. Mantissa (fraction) bits = 01000...0 → fraction = 0.25, so significand = 1 + 0.25 = 1.25.
2. R1's decimal value = 1.25 × 2^5 = 1.25 × 32 = 40.
3. Decode R2 = 0xC1200000: binary 1100 0001 0010 0000...0000. Sign = 1 (negative). Exponent bits = 10000010 = 130; unbiased exponent = 130 - 127 = 3. Mantissa fraction = 0.25, so significand = 1.25.
4. R2's decimal value = -(1.25 × 2^3) = -(1.25 × 8) = -10.
5. Compute R3 = R1 / R2 = 40 / (-10) = -4.
6. Encode -4 in IEEE-754 single precision: -4 = -1.0 × 2^2. Sign bit = 1 (negative). Exponent = 2 + 127 = 129 = binary 10000001. Since the significand is exactly 1.0, the mantissa (fraction) field is all zeros.

7. Assemble the 32 bits: 1 10000001 00000000000000000000000. Grouping into hex nibbles gives 1100 0000 1000 0000 0000 0000 0000 0000 = 0xC0800000.
8. This matches option (B).

KEY IDEA
To divide two IEEE-754 single-precision numbers, decode each into sign, exponent (unbiased), and significand, do the arithmetic in decimal, then re-encode the result back into IEEE-754 bits.

WATCH OUT FOR
Forgetting to un-bias the exponent (subtracting 127) before combining, or mis-tracking the sign bit when one operand is negative, leading to a wrong-magnitude or wrong-sign final hex encoding.

Q.30 MCQ · 2 MARKS

Computer Organization and Architecture › Memory & cache

A computer system with a word length of 32 bits has a 16 MB byte-addressable main memory and a 64 KB, 4-way set associative cache memory with a block size of 256 bytes. Consider the following four physical addresses represented in hexadecimal notation.

$A1 = 0x42C8A4,\ A2 = 0x546888,\ A3 = 0x6A289C,\ A4 = 0x5E4880$

Which one of the following is TRUE?

(A) A1 and A4 are mapped to different cache sets.

(B) A2 and A3 are mapped to the same cache set.

(C) A3 and A4 are mapped to the same cache set.

(D) A1 and A3 are mapped to the same cache set.

Official answer: **B**

SOLUTION

1. Physical address size: main memory = 16 MB = 2^24 B, so addresses are 24 bits wide.
2. Block offset: block size = 256 B = 2^8 B, so the low 8 bits (bits[7:0]) are the block offset.
3. Number of sets: cache = 64 KB = 65536 B, so number of blocks = 65536/256 = 256. With 4-way associativity, number of sets = 256/4 = 64 = 2^6, so 6 bits (bits[13:8]) form the set index. The remaining 10 bits (bits[23:14]) are the tag.
4. Compute the set index for each address as (address >> 8) & 0x3F, equivalent to reading bits[13:8] directly from the binary expansion:
5. A1 = 0x42C8A4 → binary 0100 0010 1100 1000 1010 0100 → bits[13:8] = 001000 = set 8.
6. A2 = 0x546888 → binary 0101 0100 0110 1000 1000 1000 → bits[13:8] = 101000 = set 40.
7. A3 = 0x6A289C → binary 0110 1010 0010 1000 1001 1100 → bits[13:8] = 101000 = set 40.
8. A4 = 0x5E4880 → binary 0101 1110 0100 1000 1000 0000 → bits[13:8] = 001000 = set 8.
9. Compare sets: A1 & A4 → both set 8 (same set, so option A, which claims they differ, is false). A2 & A3 → both set 40 (same set) → option B is TRUE. A3 (40) & A4 (8) differ → option C false. A1 (8) & A3 (40) differ → option D false.
10. Hence the TRUE statement is option B.

KEY IDEA

Split the 24-bit physical address (since main memory $= 16\ \text{MB} = 2^{24}\ \text{B}$) into Tag | Set Index | Block Offset. Block offset width = $\log_2(\text{block size})$. Number of cache sets = (cache size / block size) / associativity, so set-index width = $\log_2(\text{sets})$. Two addresses collide in the cache iff their set-index bits are equal, regardless of their tag bits.

WATCH OUT FOR

Miscounting the number of index bits by using $\log_2(\text{cache size}/\text{block size})$ (giving 8 bits) instead of dividing by the associativity first to get the number of SETS ($64\text{KB}/256\text{B} = 256$ blocks, $256/4 = 64$ sets $\Rightarrow$ 6 index bits), which shifts every set number and can flip which addresses collide.

Q.31 MCQ · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Let $G = (V, E)$ be a weighted undirected graph and let T be a Minimum Spanning Tree (MST) of G maintained using adjacency lists. Suppose a new weighted edge $(u, v) \in V \times V$ is added to G. The worst case time complexity of determining if T is still an MST of the resultant graph is

(A) $\Theta(|E| + |V|)$

(B) $\Theta(|E||V|)$

(C) $\Theta(|E| \log |V|)$

(D) $\Theta(|V|)$

Official answer: **D**

SOLUTION

1. T is a spanning tree of G, so it has exactly |V|-1 edges (independent of |E|), stored using adjacency lists.
2. When edge (u,v) is added, T (with one edge possibly swapped) remains an MST of the new graph if and only if w(u,v) is not less than the weight of the maximum-weight edge on the unique path between u and v in T; this follows from the MST cycle/cut property (if w(u,v) were smaller, swapping it in for that max edge would give a strictly lighter spanning tree).
3. To apply this test, find the unique u-v path in T (unique because T is a tree) and its maximum edge weight, e.g. by a BFS/DFS confined to T starting from u and stopping at v.
4. T has |V| vertices and |V|-1 edges, so this traversal over the adjacency-list representation costs Θ(|V| + (|V|-1)) = Θ(|V|).
5. None of the remaining |E| - (|V|-1) edges of G need to be examined, so the running time does not depend on |E|.
6. Hence the worst-case time complexity is Θ(|V|), which is option (D).

KEY IDEA

By the MST cycle property, adding an edge (u, v) keeps T an MST iff the new edge's weight is not less than the weight of the maximum-weight edge on the unique tree path between u and v; since T has only $|V| - 1$ edges, finding that path and its max edge by a traversal restricted to T costs $\Theta(|V|)$, independent of $|E|$.

WATCH OUT FOR

Assuming the check must scan the whole graph G (all $|E|$ edges) or rerun a full MST algorithm such as Kruskal's ($\Theta(|E| \log |V|)$), instead of noticing that only the tree T (with exactly $|V| - 1$ edges) needs to be traversed.

Q.32 MCQ · 2 MARKS

Theory of Computation › Context-free languages

Consider the following languages.

$L_1 = \{wxyx \mid w, x, y \in (0 + 1)^+\}$

$L_2 = \{xy \mid x, y \in (a+b)^*, |x| = |y|, x \neq y\}$
Which one of the following is TRUE?

(A) L_1 is regular and L_2 is context-free.

(B) L_1 is context-free but not regular and L_2 is context-free.

(C) Neither L_1 nor L_2 is context-free.

(D) L_1 is context-free but L_2 is not context-free.

Official answer: **A**

SOLUTION

1. L_1 **is regular.** Every string of L_1 can be rewritten with a one-symbol x. Write a member as $w\,x\,y\,x$ with $x = x_1 \cdots x_k$ and let $a = x_k$. Then the string is $(w x_1 \cdots x_{k-1})\ a\ (y\,x_1 \cdots x_{k-1})\ a$. The first and third blocks are nonempty, so it has the form $w'\,a\,y'\,a$ with w', y' nonempty. Conversely, a string of that form is in L_1 by taking $x = a$. So $L_1 = \{w\,a\,y\,a \mid w, y \in (0+1)^+,\ a \in \{0,1\}\}$.
2. A string of length n is in L_1 exactly when its last symbol a also occurs at some position p with $2 \le p \le n-2$. Position $p \ge 2$ leaves room for a nonempty w, and $p \le n-2$ leaves room for a nonempty y before the final a.
3. Hence $L_1 = (0+1)^+\,0\,(0+1)^+\,0\ +\ (0+1)^+\,1\,(0+1)^+\,1$. This is a regular expression, so L_1 is regular, and therefore also context-free.
4. L_2 **is context-free.** A string $z = xy$ with $|x| = |y| = n$ is in L_2 exactly when $z_p \neq z_{p+n}$ for some position $p \le n$, so z has even length $2n$. Take the grammar $S \to AB \mid BA$, $A \to XAX \mid a$, $B \to XBX \mid b$, $X \to a \mid b$. Here A generates exactly the odd-length strings over $\{a, b\}$ whose middle symbol is a, and B those whose middle symbol is b.
5. Soundness: let A produce a string of length $2i+1$ and B one of length $2j+1$. Then AB has length $2n$ with $n = i + j + 1$. The a is at position $i+1$ and the b is at position $(2i+1) + (j+1) = 2i + j + 2$. These are exactly n apart, so the two halves differ at position $i+1$. The rule BA is the same with the roles of a and b swapped.
6. Completeness: if z has length $2n$ with $z_p \neq z_{p+n}$ and $z_p = a$, split z as $z[1..2p-1]$ followed by $z[2p..2n]$. The first part has odd length and middle symbol $z_p = a$, so A derives it. The second part has length $2n - 2p + 1$ and its middle symbol is at position $p + n$, which is b, so B derives it. If instead $z_p = b$ and $z_{p+n} = a$, the same split works with BA. So the grammar generates exactly L_2, and L_2 is context-free.
7. Therefore L_1 is regular and L_2 is context-free, which is option (A). Options (B) and (D) are false because L_1 is regular (not 'not regular'), and (D) also because L_2 is context-free. Option (C) is false because both languages are context-free.

KEY IDEA

The word $wxyx$ with long x collapses to $w\,a\,y\,a$ with a single symbol a, so L_1 is regular. The set of even-length strings whose halves differ is context-free: guess one mismatch pair, which sits exactly half the length apart, using two odd-length palindrome-style blocks with centres a and b.

WATCH OUT FOR

Treating L_1 as a 'copy language' that needs memory of x and so is not regular, or thinking L_2 (with $x \neq y$) is not context-free because $\{xx\}$ is not. The complement of $\{xx\}$ is context-free because it only needs to remember one mismatch position.

Q.33 MCQ · 2 MARKS

Compiler Design › Syntax-directed translation

Consider the productions A → PQ and A → XY. Each of the five non-terminals A, P, Q, X, and Y has two attributes: s is a synthesized attribute, and i is an inherited attribute. Consider the following rules.

Rule 1: P.i = A.i + 2, Q.i = P.i + A.i, and A.s = P.s + Q.s

Rule 2: X.i = A.i + Y.s and Y.i = X.s + A.i

Which one of the following is TRUE?

(A) Both Rule 1 and Rule 2 are L-attributed.

(B) Only Rule 1 is L-attributed.

(C) Only Rule 2 is L-attributed.

(D) Neither Rule 1 nor Rule 2 is L-attributed.

Official answer: **B**

SOLUTION

1. A definition is L-attributed if, for every production $A \to X_1 X_2 \dots X_n$, each inherited attribute of X_j depends only on (i) inherited attributes of A and (ii) attributes (inherited or synthesized) of $X_1, \dots, X_{j-1}$, the symbols to its left. Synthesized attributes of A may depend on any attribute in the body.
2. Rule 1, production $A \to PQ$: $P.i = A.i + 2$ uses only the inherited attribute of the head, so it is allowed.
3. $Q.i = P.i + A.i$ uses P, which is to the left of Q, and the head's inherited attribute $A.i$, so it is allowed.
4. $A.s = P.s + Q.s$ defines a synthesized attribute of the head, which may use any child. So Rule 1 is L-attributed.
5. Rule 2, production $A \to XY$: $X.i = A.i + Y.s$ makes the inherited attribute of X depend on $Y.s$, an attribute of Y, which is to the right of X. This violates the left-to-right condition, so Rule 2 is not L-attributed.
6. $Y.i = X.s + A.i$ alone would be allowed, because X is to the left of Y. One violation is enough to fail the definition. Together, $X.i$ needs $Y.s$ and $Y.i$ needs $X.s$, so the dependencies are also circular.
7. Hence only Rule 1 is L-attributed: option (B).

KEY IDEA
In an L-attributed definition, an inherited attribute of a body symbol may use only the head's inherited attributes and attributes of symbols to its left. Never an attribute of a symbol to its right.

WATCH OUT FOR
Checking only $Y.i = X.s + A.i$, which is legal, and ignoring that $X.i = A.i + Y.s$ looks to the right. Another slip is to think a synthesized attribute like $A.s$ is restricted in the same way.

Q.34 MCQ · 2 MARKS

Operating Systems › Synchronisation

Each of a set of n processes executes the following code using two semaphores `a` and `b` initialized to 1 and 0, respectively. Assume that `count` is a shared variable initialized to 0 and not used in CODE SECTION P.

```
CODE SECTION P

wait(a); count=count+1;
if (count==n) signal(b);
signal(a); wait(b); signal(b);

CODE SECTION Q
```

What does the code achieve?

(A) It ensures that no process executes CODE SECTION Q before every process has finished CODE SECTION P.

(B) It ensures that at most two processes are in CODE SECTION Q at any time.

(C) It ensures that all processes execute CODE SECTION P mutually exclusively.

(D) It ensures that at most $n-1$ processes are in CODE SECTION P at any time.

Official answer: **A**

SOLUTION

1. Semaphore `a` starts at 1, so it acts as a mutex. It protects only the update `count=count+1` and the test `count==n`. CODE SECTION P sits before `wait(a)`, outside that lock.
2. Semaphore `b` starts at 0, so `wait(b)` blocks every process until some process executes `signal(b)`.
3. Each process increments `count` exactly once, under the lock. Only the n-th process to arrive sees `count==n`, so only that process executes the first `signal(b)`.
4. Until the n-th process arrives, `b` stays 0. Each of the first $n-1$ processes finishes P, releases `a`, and blocks at `wait(b)`.
5. After the n-th process has finished P and signalled `b`, the value of `b` is 1. One blocked process wakes, executes `signal(b)` and so passes the token on. This repeats until all n processes are released. Every process passes through `wait(b); signal(b)`, so this is a one-use barrier (a turnstile).
6. No process reaches CODE SECTION Q before `b` has been signalled. That signal happens only after the n-th process has run its increment, which comes after that process has finished P, and every other process had finished P before it incremented. So no process enters Q before every process has finished P, and (A) holds.
7. Check the others. (B) After the release, all n processes can be in Q together, so there is no limit of two. (C) P is outside the mutex, so processes can run P concurrently. (D) All n processes may be in P at the same time, since nothing limits entry to it.

KEY IDEA
A semaphore initialised to 0 that the last arriving process signals, and that each released process signals again, is a one-use barrier. The mutex only guards the arrival counter.

WATCH OUT FOR
Reading semaphore `a` as protecting CODE SECTION P. It guards only the counter update, and P runs before `wait(a)`, so option (C) is false.

Q.35 MCQ · 2 MARKS

Operating Systems › File systems & disks

Consider the following five disk access requests of the form (request id, cylinder number) that are present in the disk scheduler queue at a given time.

(P, 155), (Q, 85), (R, 110), (S, 30), (T, 115)

Assume the head is positioned at cylinder 100. The scheduler follows Shortest Seek Time First scheduling to service the requests.

Which one of the following statements is FALSE?

(A) T is serviced before P.

(B) Q is serviced after S, but before T.

(C) The head reverses its direction of movement between servicing of Q and P.

(D) R is serviced before P.

Official answer: **B**

SOLUTION

1. Shortest Seek Time First always serves the pending request whose cylinder is nearest the current head position. The head starts at cylinder 100.

2. From 100 the distances are $P = 55$, $Q = 15$, $R = 10$, $S = 70$, $T = 15$. The nearest is R (110), so serve R. The head is now at 110.
3. From 110 the distances are $P = 45$, $Q = 25$, $S = 80$, $T = 5$. Serve T (115). The head is now at 115.
4. From 115 the distances are $P = 40$, $Q = 30$, $S = 85$. Serve Q (85). The head is now at 85.
5. From 85 the distances are $P = 70$, $S = 55$. Serve S (30). The head is now at 30. The last request is P (155).
6.

Order	1	2	3	4	5
Request	R	T	Q	S	P
Cylinder	110	115	85	30	155

7. Check the options against this order. (A) T is 2nd and P is 5th, so T is served before P: true. (B) Q is 3rd and S is 4th, so Q is served before S, not after it (and it is also after T, not before it): false. (C) After Q (85) the head moves down to 30 for S, then up to 155 for P, so it reverses direction between Q and P: true. (D) R is 1st and P is 5th: true.
8. The only false statement is (B).

KEY IDEA

With SSTF, recompute the distance from the current head position to every pending request after each service and always take the smallest. Write down the full service order, then test each option against it.

WATCH OUT FOR

Computing every distance from the starting cylinder 100 once and sorting by it. That gives R, then Q or T, then P, then S, which is wrong because the head moves after each service. Another slip is missing that Q (85) and T (115) are both 15 away at the start, so the tie only matters if you do not move first. Here R is strictly nearest, and the head then reaches T before Q.

Q.36 MCQ · 2 MARKS

Database Management and Warehousing › Normal forms

Consider a relational table R that is in 3NF, but not in BCNF. Which one of the following statements is TRUE?

(A) R has a nontrivial functional dependency $X \to A$, where X is not a superkey and A is a prime attribute.

(B) R has a nontrivial functional dependency $X \to A$, where X is not a superkey and A is a non-prime attribute and X is not a proper subset of any key.

(C) R has a nontrivial functional dependency $X \to A$, where X is not a superkey and A is a non-prime attribute and X is a proper subset of some key.

(D) A cell in R holds a set instead of an atomic value.

Official answer: **A**

SOLUTION

1. R is not in BCNF, so some nontrivial FD $X \to A$ holds in R with X not a superkey.
2. R is in 3NF, so every nontrivial FD $X \to A$ must satisfy at least one of: X is a superkey, or A is a prime attribute.
3. For the FD in step 1, X is not a superkey, so the second condition must hold: A is a prime attribute. This is exactly option (A), so (A) is true.
4. Options (B) and (C) both claim a nontrivial FD $X \to A$ with X not a superkey and A non-prime. That is the case 3NF forbids, so both are false, whatever X is a subset of.
5. Option (D) is false. 3NF includes 1NF, so every cell holds an atomic value.

KEY IDEA

3NF allows an FD $X \to A$ when X is a superkey or A is prime. BCNF allows only the first. A table in 3NF but not BCNF must therefore have an FD whose determinant is not a superkey and whose right side is prime.

WATCH OUT FOR

Picking (C) because partial dependencies sound like a normal-form violation. A non-prime attribute depending on part of a key breaks 2NF, so a table in 3NF cannot have one. Another slip is choosing (D) by confusing 3NF with the 1NF requirement.

Q.37 MCQ · 2 MARKS

Database Management and Warehousing › Transactions

Q.No. 37 Consider a schedule of transactions T_1 and T_2:

T_1	RA			RC		WD		WB	Commit	
T_2		RB	WB		RD		WC			Commit

Here, RX stands for "Read(X)" and WX stands for "Write(X)". Which one of the following schedules is conflict equivalent to the above schedule?

(A)

T_1				RA	RC	WD	WB		Commit	
T_2	RB	WB	RD					WC		Commit

(B)

T_1	RA	RC	WD	WB					Commit	
T_2					RB	WB	RD	WC		Commit

(C)

T_1	RA	RC	WD				WB		Commit	
T_2				RB	WB	RD		WC		Commit

(D)

T_1					RA	RC	WD	WB	Commit	
T_2	RB	WB	RD	WC						Commit

Official answer: **A**

SOLUTION

1. Write the given schedule in time order: $RA_1, RB_2, WB_2, RC_1, RD_2, WD_1, WC_2, WB_1, C_1, C_2$.
2. Two operations conflict if they belong to different transactions, touch the same item, and at least one of them is a write. T_1 alone uses A, so A gives no conflict. Items B, C and D are touched by both transactions.
3. Item B: RB_2 and WB_2 come before WB_1. Required: both T_2 operations on B precede WB_1.
4. Item C: RC_1 comes before WC_2. Required: $RC_1 < WC_2$.
5. Item D: RD_2 comes before WD_1. Required: $RD_2 < WD_1$.
6. A schedule is conflict equivalent to the given one exactly when it keeps each transaction's own order and all three of these orders.
7. Option (A): RB_2, WB_2, RD_2 come first, then RA_1, RC_1, WD_1, WB_1, then WC_2. So $WB_2 < WB_1$, $RD_2 < WD_1$ and $RC_1 < WC_2$ all hold. Conflict equivalent.
8. Option (B): T_1 runs completely first, so $WB_1 < RB_2$. This reverses the order on B. Not equivalent.

9. Option (C): WD_1 comes before RD_2. This reverses the order on D. Not equivalent.
10. Option (D): WC_2 comes before RC_1. This reverses the order on C. Not equivalent. Hence only (A) is conflict equivalent.

KEY IDEA
List every pair of conflicting operations (different transactions, same item, at least one write) in the given schedule. A candidate is conflict equivalent only if it keeps the relative order of every such pair.

WATCH OUT FOR
Comparing only the order of the transactions as a whole and ignoring item-by-item conflicts. Each wrong option reverses exactly one conflicting pair: B reverses the B pair, C the D pair, D the C pair.

Q.38 MCQ · 2 MARKS

Computer Networks › IPv4

An organization requires a range of IP addresses to assign one to each of its 1500 computers. The organization has approached an Internet Service Provider (ISP) for this task. The ISP uses CIDR and serves the requests from the available IP address space 202.61.0.0/17. The ISP wants to assign an address space to the organization which will minimize the number of routing entries in the ISP's router using route aggregation. Which of the following address spaces are potential candidates from which the ISP can allot any one to the organization?

I. 202.61.84.0/21
II. 202.61.104.0/21
III. 202.61.64.0/21
IV. 202.61.144.0/21

(A) I and II only
(B) II and III only
(C) III and IV only
(D) I and IV only

Official answer: **B**

SOLUTION

1. Hosts needed: $1500 \leq 2^{11} = 2048$ and $2^{10} = 1024 < 1500$, so 11 host bits are needed and the prefix length is $32 - 11 = 21$. Every candidate is a /21, which matches (a /21 holds 2048 addresses, 2046 of them usable, enough for 1500 computers).
2. The ISP's pool 202.61.0.0/17 has 17 network bits: the first two octets (16 bits) plus the top bit of the third octet, which is fixed at 0. The other 7 bits of the third octet and all of the fourth octet are free, so the pool spans 202.61.0.0 to 202.61.127.255, i.e. third octet 0 to 127.
3. To keep the ISP's routing table small by route aggregation, the allotted block must lie inside the ISP's /17, so that the ISP still advertises one /17 entry for it. The block must also be a valid /21 network address: a /21 leaves 3 bits of the third octet for hosts, so the third octet must be a multiple of $2^3 = 8$.
4. I. 202.61.84.0/21: $84 = 10 \times 8 + 4$, not a multiple of 8, so 202.61.84.0 is not a /21 network address (the /21 containing it starts at 202.61.80.0). Not valid.
5. II. 202.61.104.0/21: $104 = 13 \times 8$ and $104 < 128$. Valid and inside the pool. It covers 202.61.104.0 to 202.61.111.255.
6. III. 202.61.64.0/21: $64 = 8 \times 8$ and $64 < 128$. Valid and inside the pool. It covers 202.61.64.0 to 202.61.71.255.
7. IV. 202.61.144.0/21: $144 = 18 \times 8$ is a valid /21, but $144 \geq 128$, so it lies outside 202.61.0.0/17 and is not part of the ISP's pool. Not available.
8. Only II and III qualify: option B.

KEY IDEA
A block can be allotted from the ISP's pool only if it is a properly aligned /21 (third octet a multiple of 8) and lies inside 202.61.0.0/17 (third octet 0 to 127). Then it aggregates into the ISP's single /17 route.

WATCH OUT FOR
Checking only the range and accepting 202.61.84.0/21, which is not aligned to a /21 boundary (84 is not a multiple of 8). Or checking only alignment and accepting 202.61.144.0/21, which is aligned but outside the /17 pool (third octet 144 is at least 128).

Q.39 MCQ · 2 MARKS

Discrete Mathematics › Mathematical logic

Which one of the following predicate formulae is NOT logically valid?

Note that W is a predicate formula without any free occurrence of x.

(A) $\forall x(p(x) \vee W) \equiv \forall x\, p(x) \vee W$

(B) $\exists x(p(x) \wedge W) \equiv \exists x\, p(x) \wedge W$

(C) $\forall x(p(x) \rightarrow W) \equiv \forall x\, p(x) \rightarrow W$

(D) $\exists x(p(x) \rightarrow W) \equiv \forall x\, p(x) \rightarrow W$

Official answer: **C**

SOLUTION

1. Since x does not occur free in W, a quantifier over x can be moved across W for $\wedge$ and $\vee$. So $\forall x(p(x) \vee W) \equiv \forall x\, p(x) \vee W$ and $\exists x(p(x) \wedge W) \equiv \exists x\, p(x) \wedge W$. Options (A) and (B) are logically valid.
2. Rewrite the implication: $p(x) \rightarrow W \equiv \neg p(x) \vee W$.
3. Option (D): $\exists x(\neg p(x) \vee W) \equiv \exists x\, \neg p(x) \vee W \equiv \neg \forall x\, p(x) \vee W \equiv \forall x\, p(x) \rightarrow W$. So (D) is valid.
4. Option (C): $\forall x(\neg p(x) \vee W) \equiv \forall x\, \neg p(x) \vee W \equiv \neg \exists x\, p(x) \vee W \equiv \exists x\, p(x) \rightarrow W$. This is not equivalent to $\forall x\, p(x) \rightarrow W$, because $\exists$ and $\forall$ differ.
5. Counter-model for (C): take the domain $\{1, 2\}$ with $p(1)$ true, $p(2)$ false and W false. Left side: $\forall x(p(x) \rightarrow W)$ is false, because at $x = 1$ we get true $\rightarrow$ false, which is false. Right side: $\forall x\, p(x)$ is false, so $\forall x\, p(x) \rightarrow W$ is true. The two sides have different truth values, so the equivalence in (C) is not valid.
6. Hence the formula that is NOT logically valid is (C).

KEY IDEA
A quantifier moves freely across a formula W in which x is not free when the connective is $\wedge$ or $\vee$. Under $\rightarrow$, the quantifier flips when it passes into the antecedent: $\forall x(p(x) \rightarrow W) \equiv \exists x\, p(x) \rightarrow W$ and $\exists x(p(x) \rightarrow W) \equiv \forall x\, p(x) \rightarrow W$.

WATCH OUT FOR
Pulling $\forall$ out of the antecedent of an implication unchanged. $\forall x(p(x) \rightarrow W)$ is not $\forall x\, p(x) \rightarrow W$. The antecedent turns the universal quantifier into an existential one.

Q.40 MCQ · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Let $G = (V, E)$ be a directed, weighted graph with weight function $w : E \rightarrow \mathbb{R}$.

For some function $f : V \rightarrow \mathbb{R}$, for each edge $(u, v) \in E$, define $w'(u, v)$ as

$$w(u, v) + f(u) - f(v)$$

Which one of the options completes the following sentence so that it is TRUE?

"The shortest paths in G under w are shortest paths under w' too, ___."

(A) for every $f : V \rightarrow \mathbb{R}$

(B) if and only if $\forall u \in V$, $f(u)$ is positive

(C) if and only if $\forall u \in V$, $f(u)$ is negative

(D) if and only if $f(u)$ is the distance from s to u in the graph obtained by adding a new vertex s to G and edges of zero weight from s to every vertex of G

Official answer: **A**

SOLUTION

1. Take any directed path P from a fixed vertex u to a fixed vertex v, passing through u = x_0, x_1, ..., x_k = v.
2. By definition of w', its total weight along P is w'(P) = sum_{i=0}^{k-1} (w(x_i,x_{i+1}) + f(x_i) - f(x_{i+1})).
3. The f-terms telescope: sum_{i=0}^{k-1} (f(x_i) - f(x_{i+1})) = f(x_0) - f(x_k) = f(u) - f(v), since every intermediate f(x_i) cancels once with + and once with -.
4. So w'(P) = w(P) + (f(u) - f(v)) for EVERY path P from u to v -- the extra term depends only on the endpoints u, v, never on which path or intermediate vertices are chosen.
5. Adding the same constant f(u)-f(v) to the weight of every candidate u-to-v path is an order-preserving shift: it can never change which path(s) among them have minimum total weight, regardless of the sign of f(u)-f(v).
6. Hence the shortest path between any fixed pair of vertices is identical under w and under w'.
7. Nothing in this argument restricted f to being positive, negative, or a specific distance function -- it holds for every f : V → R, matching option (A) and ruling out the unnecessary conditions in (B), (C) and (D).

KEY IDEA

Reweighting every edge (u,v) by w'(u,v) = w(u,v) + f(u) - f(v) changes the total weight of ANY path P from a fixed u to a fixed v by exactly the same constant f(u) - f(v) (the intermediate f-terms telescope away), regardless of which internal vertices P visits. Since a constant additive shift applied identically to every competing u-to-v path can never change which of them has the smallest total weight, the set of shortest paths between any pair of vertices is identical under w and under w' -- for every choice of f, with no sign restriction and no special construction required.

WATCH OUT FOR

Students assume Johnson's algorithm's specific choice of f (the shortest-distance-from-an-added-source construction, option D) is necessary for shortest paths to be preserved, because that is the f used in practice. In fact that construction is only needed to make w' nonnegative so Dijkstra becomes usable -- a separate, additional goal. Preservation of shortest paths itself holds for every f with no restriction, since the endpoint-only additive shift is order-preserving regardless of its sign.

Q.41 MCQ · 2 MARKS

Data Structures and Algorithms › Trees & heaps

In a balanced binary search tree with n elements, what is the worst case time complexity of reporting all elements in range $[a, b]$? Assume that the number of reported elements is k.

(A) $\Theta(\log n)$

(B) $\Theta(\log n + k)$

(C) $\Theta(k \log n)$

(D) $\Theta(n \log k)$

Official answer: **B**

Q.42 NAT · 2 MARKS

Discrete Mathematics › Counting

The number of permutations of the characters in LILAC so that no character appears in its original position,

if the two L's are indistinguishable, is _____.

Official answer: **12**

SOLUTION

1. In LILAC the characters sit at positions $1, 2, 3, 4, 5$ as L, I, L, A, C. So L is originally at positions 1 and 3, I at 2, A at 4 and C at 5. The two L's are identical, so the condition on them is that neither L may sit in position 1 or 3.
2. Hence the two L's must occupy two of the positions $\{2, 4, 5\}$. There are $\binom{3}{2} = 3$ choices. The remaining three positions are filled by I, A, C in some order, and each of these three letters keeps its own restriction.
3. Case L at $\{2, 4\}$: the free positions are $1, 3, 5$. Positions 2 and 4 (the original places of I and A) are occupied, so I and A are unrestricted. Only C must avoid position 5. Count: $3! - 2! = 6 - 2 = 4$.
4. Case L at $\{2, 5\}$: the free positions are $1, 3, 4$. Position 2 is occupied, so I is unrestricted. Position 5 is occupied, so C is unrestricted. Only A must avoid position 4. Count: $3! - 2! = 4$.
5. Case L at $\{4, 5\}$: the free positions are $1, 2, 3$. A and C are unrestricted. Only I must avoid position 2. Count: $3! - 2! = 4$.
6. The three cases are disjoint, because they place the L's differently. Total $= 4 + 4 + 4 = 12$.

KEY IDEA

Identical letters share a single forbidden set. Both L's must avoid positions 1 and 3, so they must occupy two of the remaining three positions. Then count the arrangements of the other letters case by case, using complement counting for the one restricted letter in each case.

WATCH OUT FOR

Treating the two L's as distinct and then dividing by 2 (or applying the derangement count for 5 distinct objects) gives the wrong number. Another slip is forgetting that I and A lose their restriction when their original positions are taken by L's.

Q.43 NAT · 2 MARKS

Computer Organization and Architecture › Pipelining

Consider a non-pipelined processor operating at 2.5 GHz. It takes 5 clock cycles to complete an instruction. You are going to make a 5-stage pipeline out of this processor. Overheads associated with pipelining force you to operate the pipelined processor at 2 GHz. In a given program, assume that 30% are memory instructions, 60% are ALU instructions and the rest are branch instructions. 5% of the memory instructions cause stalls of 50 clock cycles each due to cache misses and 50% of the branch instructions cause stalls of 2 cycles each. Assume that there are no stalls associated with the execution of ALU instructions. For this program, the speedup achieved by the pipelined processor over the non-pipelined processor (round off to 2 decimal places) is ____.

Official answer: **2.15 to 2.18**

SOLUTION

1. Non-pipelined processor: clock = 2.5 GHz ⟹ cycle time = 1/2.5 = 0.4 ns. CPI = 5 (given). Time per instruction = 5 × 0.4 ns = 2 ns.
2. Pipelined processor: clock = 2 GHz ⟹ cycle time = 1/2 = 0.5 ns. Ideal pipeline CPI (no stalls) = 1.
3. Instruction mix: memory = 30%, ALU = 60%, branch = 100% − 30% − 60% = 10%.
4. Average stall cycles/instruction from memory misses = 0.30 (fraction memory) × 0.05 (fraction of those that miss) × 50 (stall cycles) = 0.75.
5. Average stall cycles/instruction from branch stalls = 0.10 (fraction branch) × 0.50 (fraction of those that stall) × 2 (stall cycles) = 0.10. ALU instructions contribute 0 stall cycles (given).
6. Total average stall cycles per instruction = 0.75 + 0.10 = 0.85. Effective pipelined CPI = 1 + 0.85 = 1.85.

7. Time per instruction, pipelined = 1.85 × 0.5 ns = 0.925 ns.
8. Speedup = time per instruction (non-pipelined) / time per instruction (pipelined) = 2 ns / 0.925 ns = 2.1621... ≈ 2.16.

KEY IDEA

Speedup from pipelining = (time per instruction, non-pipelined) / (time per instruction, pipelined), where each "time per instruction" = CPI × clock cycle time, and the pipelined CPI must include average stall cycles contributed by each instruction class weighted by its frequency and its own stall probability.

WATCH OUT FOR

Forgetting that the branch instruction fraction is the remainder (100% − 30% − 60% = 10%), not 30% again, or adding the base CPI of 1 after computing average stall cycles incorrectly instead of before multiplying by cycle time; also mixing up which processor's cycle time (0.4 ns vs 0.5 ns) goes with which CPI.

Q.44 NAT · 2 MARKS

Computer Organization and Architecture › Instructions & addressing

A processor has 64 registers and uses 16-bit instruction format. It has two types of instructions: I-type and R-type. Each I-type instruction contains an opcode, a register name, and a 4-bit immediate value. Each R-type instruction contains an opcode and two register names. If there are 8 distinct I-type opcodes, then the maximum number of distinct R-type opcodes is ___.

Official answer: **14**

SOLUTION

1. Register field width: with 64 registers, a register name needs $\log_2 64 = 6$ bits.
2. I-type opcode length: an I-type instruction packs an opcode, one register name, and a 4-bit immediate into 16 bits, so the opcode field is $16 - 6 - 4 = 6$ bits long.
3. R-type opcode length: an R-type instruction packs an opcode and two register names into 16 bits, so the opcode field is $16 - 6 - 6 = 4$ bits long.
4. Since I-type and R-type opcodes are read from the same leading bits of a 16-bit word, no R-type opcode (4 bits) may be a bit-prefix of any I-type opcode (6 bits), and vice versa; the full set of opcodes must be a prefix-free (instantaneous) code, exactly as in Huffman coding.
5. Model all possible opcode bit-strings as leaves of a binary tree of depth 6 (the longest opcode length): there are $2^6 = 64$ leaf slots in total.
6. Each of the 8 chosen I-type opcodes is a full-length (6-bit) codeword, so it occupies exactly 1 leaf slot each: $8 \times 1 = 8$ slots used.
7. Each R-type opcode is only 4 bits long, so to keep the code prefix-free it must reserve every one of its length-6 extensions: $2^{6-4} = 4$ leaf slots per R-type opcode.
8. Leaf slots left after removing the I-type opcodes: $64 - 8 = 56$.
9. Maximum number of distinct R-type opcodes: $56/4 = 14$ (this is the same bound as Kraft's inequality $8 \cdot 2^{-6} + n \cdot 2^{-4} \leq 1 \Rightarrow n \leq 14$).

KEY IDEA

Because opcodes of two different bit-lengths share one fixed-width instruction word, the decoder must be able to tell them apart just by reading bits left to right: the set of all opcodes must form a prefix-free (instantaneous) code, exactly as in Huffman/variable-length coding. Kraft's inequality $\sum 2^{-\ell_i} \le 1$ (equivalently, counting reserved leaf slots in a binary code-tree) then bounds how many short codewords can coexist with the given long ones.

WATCH OUT FOR

Treating the R-type opcode field size (4 bits) as directly giving $2^4 = 16$ possible R-type opcodes, ignoring that some of those 4-bit patterns must be excluded because they would otherwise be indistinguishable, as prefixes, from the 6-bit I-type opcodes sharing the same instruction word.

Q.45 NAT · 2 MARKS

Engineering Mathematics › Probability & statistics

For $n > 2$, let $a \in \{0,1\}^n$ be a non-zero vector. Suppose that x is chosen uniformly at random from $\{0,1\}^n$. Then, the probability that $\sum_{i=1}^{n} a_i x_i$ is an odd number is ____.

Official answer: **0.5**

SOLUTION

1. Only the coordinates with $a_i = 1$ matter. Let $S = \{i : a_i = 1\}$ and $k = |S| \ge 1$, since a is non-zero.
2. Then $\sum_{i=1}^{n} a_i x_i = \sum_{i \in S} x_i$, the number of ones among the k bits x_i with $i \in S$.
3. The bits are independent and uniform, so the number of ones is Binomial$(k, \frac{1}{2})$ and $P(\text{odd}) = \dfrac{\binom{k}{1} + \binom{k}{3} + \cdots}{2^k}$.
4. Use $(1+1)^k = \sum_j \binom{k}{j}$ and $(1-1)^k = \sum_j (-1)^j \binom{k}{j} = 0$ for $k \ge 1$. Subtracting the second from the first gives $2\sum_{j \text{ odd}} \binom{k}{j} = 2^k$, so $\sum_{j \text{ odd}} \binom{k}{j} = 2^{k-1}$.
5. So $P(\text{odd}) = \dfrac{2^{k-1}}{2^k} = \dfrac{1}{2}$.
6. Another way to see it: fix x_j for one $j \in S$ and let the other bits take any values. Flipping x_j flips the parity of the sum, so exactly one of its two values makes the sum odd. This pairs up the 2^n vectors into equal halves.
7. The answer is $\frac{1}{2}$ for every non-zero a, so it does not depend on n or on a.

KEY IDEA

The sum of a non-empty set of independent fair bits is odd with probability one half. Flipping any one included bit flips the parity, which pairs up the outcomes into equal halves.

WATCH OUT FOR

Thinking the answer depends on n or on how many ones a has, or counting only the all-zero vector as even. The even outcomes are not just one case. Exactly half of all x give an even sum.

Q.46 NAT · 2 MARKS

Programming in C › Recursion

Consider the following C functions.

```
int fun1(int n) {
 static int i = 0;
 if (n > 0) {
   ++i;
   fun1(n-1);
 }
 return(i);
}
```

```
int fun2(int n) {
 static int i = 0;
 if (n > 0) {
   i = i + fun1(n);
   fun2(n-1);
 }
 return(i);
}
```

The return value of `fun2(5)` is _____.

Official answer: **55**

SOLUTION

1. fun1's static variable (call it j) and fun2's static variable (call it i) are each shared across ALL calls to that respective function; they are initialized once, not per call.
2. fun1(n) with n>0 increments j once then calls fun1(n-1); the base case (n=0) just returns j. So a call fun1(n) increments j exactly n times before returning it (in decreasing order n, n-1, ..., 1).
3. Trace fun1's static j (starts at 0) across the successive top-level calls made by fun2: fun1(5) increments j by 5 → j=5, returns 5. fun1(4) then increments j by 4 more → j=9, returns 9. fun1(3) adds 3 more → j=12, returns 12. fun1(2) adds 2 more → j=14, returns 14. fun1(1) adds 1 more → j=15, returns 15.
4. Trace fun2's static i (starts at 0): fun2(5): i = 0 + fun1(5) = 0+5 = 5, then calls fun2(4).
5. fun2(4): i = 5 + fun1(4) = 5+9 = 14, then calls fun2(3).
6. fun2(3): i = 14 + fun1(3) = 14+12 = 26, then calls fun2(2).
7. fun2(2): i = 26 + fun1(2) = 26+14 = 40, then calls fun2(1).
8. fun2(1): i = 40 + fun1(1) = 40+15 = 55, then calls fun2(0), whose base case (n=0) just returns i = 55 unchanged.
9. Because every level's `return(i)` executes only after its recursive call has already returned, and nothing after the recursive call modifies i, every level of fun2, including the outermost fun2(5), returns the same final shared value, 55.

KEY IDEA

Each function has its OWN static variable, shared across every recursive call to that function (not per-call). `fun1`'s static `i` accumulates once per call to `fun1`, so `fun1(n)` returns the running total after `n` more increments. `fun2`'s static `i` accumulates `fun1(k)`'s return value at each level and is returned unchanged by every subsequent (deeper) `fun2` call, since nothing decrements it afterward.

WATCH OUT FOR

Treating `fun1`'s static `i` as reset to 0 on every fresh call from `fun2` (as if it were an ordinary local variable), which makes each `fun1(k)` call return only `k` instead of the running cumulative total. This undercounts every term after the first and gives an answer far smaller than 55.

Q.47 NAT · 2 MARKS

Data Structures and Algorithms › Trees & heaps

Consider the array representation of a binary min-heap containing 1023 elements. The minimum number of comparisons required to find the maximum in the heap is ___.

Official answer: **511**

SOLUTION

1. Min-heap property: every parent's value is ≤ its children's values. Therefore no internal (non-leaf) node can be the maximum: following any root-to-leaf path, values are non-decreasing, so the maximum overall must occur at some leaf.

2. For n = 1023 = 2^10 - 1 elements, the array-represented heap is a perfect binary tree with 10 levels (height 9).
3. In a 1-indexed array representation of a heap of size n, the leaves occupy indices floor(n/2)+1 through n, so the number of leaves is n - floor(n/2) = ceil(n/2). For n = 1023: ceil(1023/2) = 1023 - 511 = 512.
4. The min-heap property gives no ordering information among values in different leaf positions (only along root-to-leaf paths), so the maximum could be any one of these 512 leaves: all 512 must be compared against one another to determine which is largest.
5. Finding the maximum of a set of k elements by pairwise comparison requires exactly k-1 comparisons in the minimum case (a tournament/linear-scan elimination: each comparison rules out at most one candidate, and k-1 candidates must be ruled out).
6. With k = 512 leaves, the minimum number of comparisons required = 512 - 1 = 511.

KEY IDEA

In a min-heap, every parent is ≤ its children, so the maximum element can only be among the leaf nodes (an internal node always has a descendant chain going down to a leaf with values ≥ it, so it can never be the largest). Finding the maximum of any set of k unordered elements needs exactly $k-1$ comparisons in the best case (tournament method), and no heap-order information helps distinguish among the leaves since min-heap order says nothing about relative order among sibling subtrees.

WATCH OUT FOR

Assuming the maximum can be found using standard heap operations, or forgetting that in a min-heap the maximum could be any one of the leaf nodes, leading to miscounting the leaves (e.g., using floor(n/2) instead of ceil(n/2)) or forgetting the k–1 lower bound for finding a maximum among unordered elements.

Q.48 NAT · 2 MARKS

Programming in C › Pointers & arrays

Consider the following C functions.

```
int tob(int b, int* arr) {
  int i;
  for (i=0; b>0; i++) {
    if (b%2) arr[i]=1;
    else     arr[i]=0;
    b = b/2;
  }
  return (i);
}
```

```
int pp(int a, int b) {
  int arr[20];
  int i, tot = 1, ex, len;
  ex = a;
  len = tob(b,arr);
  for (i=0; i<len; i++) {
    if (arr[i]==1)
      tot = tot * ex;
    ex = ex * ex;
  }
  return (tot);
}
```

The value returned by pp(3,4) is _____.

Official answer: **81**

SOLUTION

1. tob(b,arr) writes the binary digits of b into arr[0..], LSB first: arr[i]=b%2 then b=b/2 each iteration, and returns the final value of the loop counter i (the bit count), since the for-loop tests `b>0` after each increment.
2. Trace tob(4,arr): i=0,b=4: 4%2=0 → arr[0]=0, b=2. i=1,b=2: 2%2=0 → arr[1]=0, b=1. i=2,b=1: 1%2=1 → arr[2]=1, b=0. i becomes 3; condition b>0 is false, loop exits. Returns i=3, so len=3 and arr=[0,0,1].
3. pp(3,4): ex=a=3, tot=1, len=3 (from the trace above).
4. Loop i=0: arr[0]=0, so tot is unchanged (tot=1); then ex=ex*ex=3*3=9.
5. Loop i=1: arr[1]=0, so tot is unchanged (tot=1); then ex=ex*ex=9*9=81.
6. Loop i=2: arr[2]=1, so tot=tot*ex=1*81=81; then ex=ex*ex=81*81=6561 (computed but never read again).
7. Loop ends since i=3=len. Function returns tot=81.
8. This is exponentiation-by-squaring: pp(a,b) computes a^b using the binary representation of b (here $b = 4 = 100_2$), and indeed $3^4 = 81$.

KEY IDEA

`tob(b,arr)` writes the binary representation of `b` into `arr[]`, least-significant bit first, and returns the bit count. `pp(a,b)` then uses those bits to compute a^b by repeated squaring: `ex` holds a^{2^i} at loop iteration `i`, and it is multiplied into `tot` exactly when bit `i` of `b` is 1. This is the standard fast-exponentiation-by-squaring algorithm, so `pp(a,b)` returns a^b.

WATCH OUT FOR

Forgetting that `tob` stores bits LSB-first (not MSB-first) and pairing the wrong bit with the wrong power of `ex`; or applying the final unused `ex = ex*ex` update after the loop has already produced `tot`, mistakenly squaring it again. Also common: stopping the trace of `tob(4,arr)` at `i=2` instead of correctly returning `i=3` (the increment happens before the loop condition is re-tested to `false`).

Q.49 NAT · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Consider a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_{100}\}$, $E = \{(v_i, v_j) \mid 1 \leq i < j \leq 100\}$, and weight of the edge (v_i, v_j) is $|i - j|$. The weight of minimum spanning tree of G is _____.

Official answer: **99**

SOLUTION

1. A spanning tree on |V| = 100 vertices has exactly 100 - 1 = 99 edges (standard property of trees).
2. Every edge (v_i, v_j) has weight $|i - j| \geq 1$, since i and j are distinct integers.
3. Therefore any spanning tree of G has total weight $\geq 99 \times 1 = 99$: this is a lower bound on the MST weight.
4. Consider the path $v_1 - v_2 - v_3 - \cdots - v_{100}$. It has exactly 99 edges, one between each pair of consecutive vertices, and each such edge (v_i, v_{i+1}) has weight $|i - (i+1)| = 1$.
5. This path connects all 100 vertices and, being a simple path, contains no cycle, so it is a valid spanning tree of G with total weight exactly $99 \times 1 = 99$.
6. Since 99 is both a proven lower bound on every spanning tree and is achieved by this explicit spanning tree, the minimum spanning tree weight of G is 99.

KEY IDEA
G is a complete graph on 100 vertices with edge weight |i-j|; since every edge weight is a positive integer ≥ 1, any spanning tree (which needs exactly n-1 = 99 edges) has weight ≥ 99, and the path v_1-v_2-...-v_100 (using only consecutive-index edges of weight 1) achieves this bound, so it is the MST with weight 99.

WATCH OUT FOR
Trying to compute the MST by only considering "adjacent index" edges without proving optimality, or miscounting edges (using 100 instead of 99), or assuming the MST must use edges of weight exactly matching some formula like sum of 1 to 99 rather than recognizing all path edges have weight exactly 1.

Q.50 NAT · 2 MARKS

Operating Systems › CPU scheduling

Consider the following set of processes, assumed to have arrived at time 0. Consider the CPU scheduling algorithms Shortest Job First (SJF) and Round Robin (RR). For RR, assume that the processes are scheduled in the order P_1, P_2, P_3, P_4.

Processes	P_1	P_2	P_3	P_4
Burst time (in ms)	8	7	2	4

If the time quantum for RR is 4 ms, then the absolute value of the difference between the average turnaround times (in ms) of SJF and RR (round off to 2 decimal places) is _____.

Official answer: **5.25**

SOLUTION

1. All four processes arrive at time 0, so turnaround time equals completion time.
2. SJF (non-preemptive, all arrive together) runs in order of burst time: $P_3(2), P_4(4), P_2(7), P_1(8)$. Completion times: $P_3 = 2$, $P_4 = 2 + 4 = 6$, $P_2 = 6 + 7 = 13$, $P_1 = 13 + 8 = 21$.
3. $$\text{Avg TAT}_{SJF} = \frac{2 + 6 + 13 + 21}{4} = \frac{42}{4} = 10.5$$
4. RR with $q = 4$ ms, queue order P_1, P_2, P_3, P_4. Gantt chart:

P_1	P_2	P_3	P_4	P_1	P_2
0-4	4-8	8-10	10-14	14-18	18-21

5. After the first round, P_1 has $8 - 4 = 4$ ms left and P_2 has $7 - 4 = 3$ ms left. P_3 (2 ms) finishes at 10 and P_4 (4 ms) finishes at 14. In the second round P_1 runs 14 to 18 and finishes, then P_2 runs 18 to 21 and finishes.
6. $$\text{Avg TAT}_{RR} = \frac{18 + 21 + 10 + 14}{4} = \frac{63}{4} = 15.75$$
7. Difference: $|10.5 - 15.75| = 5.25$ ms.

KEY IDEA
With all arrivals at time 0, turnaround time equals completion time. Build the SJF order by burst length and trace the RR queue quantum by quantum, then compare the two averages.

WATCH OUT FOR
Forgetting that a process with less than one quantum left (P3 with 2 ms) gives up the CPU early, which shifts every later start time. Another slip is using waiting time instead of turnaround time.

Q.51 NAT · 2 MARKS

Theory of Computation › Finite automata

Consider the following language.

$L = \{x \in \{a, b\}^* \mid \text{number of } a\text{'s in } x \text{ is divisible by 2 but not divisible by 3}\}$

The minimum number of states in a DFA that accepts L is _____.

Official answer: **6**

SOLUTION

1. Only the number of a's matters: a b leaves the situation unchanged. So it is enough to track $n_a \bmod 6$, because this fixes both $n_a \bmod 2$ and $n_a \bmod 3$ (6 = 2 x 3, coprime).
2. A string is in L exactly when $n_a \bmod 6 \in \{2, 4\}$. Residue 0 is divisible by 3, so it is excluded. Residues 1, 3 and 5 are odd, so they are excluded. Residues 2 and 4 are even and not divisible by 3.
3. So a 6-state DFA works: states $q_0, \ldots, q_5$, $\delta(q_i, a) = q_{(i+1) \bmod 6}$, $\delta(q_i, b) = q_i$, start state q_0, accepting states $\{q_2, q_4\}$.
4. Minimality: from q_i, the string a^k is accepted iff $(i + k) \bmod 6 \in \{2, 4\}$. So the set of $k \in \{0, \ldots, 5\}$ accepted from q_i is $\{2 - i,\ 4 - i\} \bmod 6$.
5. For $i = 0, \ldots, 5$ these sets are $\{2, 4\}, \{1, 3\}, \{0, 2\}, \{5, 1\}, \{4, 0\}, \{3, 5\}$. All six are different, so for any two states some a^k is accepted from one and rejected from the other. Every pair of states is distinguishable.
6. No two states can be merged. The DFA is complete (every state has a transition on both symbols), so no extra dead state is needed. The minimum number of states is 6.

KEY IDEA

Divisibility by 2 and by 3 combine into a single counter modulo $\operatorname{lcm}(2, 3) = 6$. The residues $\{2, 4\}$ are the accepting ones, and a Myhill-Nerode argument with strings a^k shows all 6 residues are distinguishable.

WATCH OUT FOR

Building a product of a mod-2 machine (2 states) and a mod-3 machine (3 states) and then guessing 5 or 2 x 3 minus something, or assuming some states can be merged. Here, because 2 and 3 are coprime, the product has 6 reachable states and none of them are equivalent.

Q.52 NAT · 2 MARKS

Discrete Mathematics › Graph theory

Graph G is obtained by adding vertex s to $K_{3,4}$ and making s adjacent to every vertex of $K_{3,4}$. The minimum number of colours required to edge-colour G is ____.

Official answer: **7**

SOLUTION

1. G has 8 vertices: the parts $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3, b_4\}$ of $K_{3,4}$, and s joined to all seven of them.
2. Degrees: $\deg s = 7$, $\deg a_i = 4 + 1 = 5$, $\deg b_j = 3 + 1 = 4$. So $\Delta(G) = 7$.
3. Lower bound: the 7 edges at s share the endpoint s, so they need 7 distinct colours. Hence $\chi'(G) \geq \Delta(G) = 7$.
4. Upper bound: first edge-colour $K_{3,4}$ with the four colours $\{0, 1, 2, 3\}$ by giving $a_i b_j$ the colour $(i + j) \bmod 4$. At a_i the four edges get $i + 1, i + 2, i + 3, i + 4 \pmod 4$, which are all different. At b_j the three edges get $j + 1, j + 2, j + 3 \pmod 4$, which are all different. So this colouring is proper.
5. Vertex b_j uses $j + 1, j + 2, j + 3 \pmod 4$ and misses colour $j \bmod 4$. Give sb_j that colour. Then $sb_1, \ldots, sb_4$ get the four different colours $1, 2, 3, 0$, and each b_j still has all its edges in different colours.
6. Each a_i already uses all of $0, 1, 2, 3$. Colour sa_1, sa_2, sa_3 with the three new colours $4, 5, 6$. These differ from each other and from the colours $0, 1, 2, 3$ on the edges sb_j, so the colours at s are all different. Each a_i sees its four old colours plus one new colour, so it is also proper.
7. This is a proper edge-colouring with 7 colours, so $\chi'(G) \leq 7$. Together with the lower bound, $\chi'(G) = 7$.

KEY IDEA
The vertex s has degree 7, which forces at least 7 colours. Then build a 7-colouring directly. Colour $K_{3,4}$ with 4 colours so that each b_j misses a different colour. Give those missing colours to the edges sb_j. The three edges sa_i take three new colours.

WATCH OUT FOR
Using $\Delta + 1 = 8$ because G is not bipartite. Vizing's theorem only says χ' is Δ or $\Delta + 1$. Here an explicit colouring shows Δ colours are enough, so G is Class 1.

Q.53 NAT · 2 MARKS

Operating Systems › Virtual memory

Consider a paging system that uses 1-level page table residing in main memory and a TLB for address translation. Each main memory access takes 100 ns and TLB lookup takes 20 ns. Each page transfer to/from the disk takes 5000 ns. Assume that the TLB hit ratio is 95%, page fault rate is 10%. Assume that for 20% of the total page faults, a dirty page has to be written back to disk before the required page is read in from disk. TLB update time is negligible. The average memory access time in ns (round off to 1 decimal places) is _____.

Official answer: **154.5 to 155.5**

SOLUTION

1. A TLB hit costs the TLB lookup plus one memory access: $20 + 100 = 120$ ns.
2. A TLB miss with the page in memory costs the TLB lookup, one access to the 1-level page table in memory, and one access for the data: $20 + 100 + 100 = 220$ ns.
3. A page that is not in memory cannot have a TLB entry, so a page fault can happen only on a TLB miss. We therefore read the 10% page fault rate as applying to TLB misses.
4. Page fault service time: with a clean victim there is one disk transfer, 5000 ns. With a dirty victim there are two transfers (write-back, then read), $2 \times 5000 = 10000$ ns. The average is $0.8 \times 5000 + 0.2 \times 10000 = 4000 + 2000 = 6000$ ns. The TLB update time is negligible.
5. Cost of a TLB miss: $220 + 0.1 \times 6000 = 220 + 600 = 820$ ns.
6. $$\text{EMAT} = 0.95 \times 120 + 0.05 \times 820 = 114 + 41 = 155 \text{ ns}$$
7. So the average memory access time is 155.0 ns.

KEY IDEA
Page faults happen only on TLB misses. Weight the hit path (120 ns) by 0.95 and the miss path (220 ns plus 0.1 times the average fault service time of 6000 ns) by 0.05.

WATCH OUT FOR
Applying the 10% fault rate to every access, or charging the fault service to TLB hits. That gives about 725 ns instead of 155 ns. Another slip is counting only one disk transfer for a dirty page, when it needs a write-back and a read.

Q.54 NAT · 2 MARKS

Database Management and Warehousing › Indexing

Consider a database implemented using B+ tree for file indexing and installed on a disk drive with block size of 4 KB. The size of search key is 12 bytes and the size of tree/disk pointer is 8 bytes. Assume that the database has one million records. Also assume that no node of the B+ tree and no records are present initially in main memory. Consider that each record fits into one disk block. The minimum number of disk accesses required to retrieve any record in the database is _____.

Official answer: **4**

SOLUTION

1. Block size $= 4\,\text{KB} = 4096$ bytes. An internal node of order p holds p tree pointers and $p - 1$ search keys.

2. Largest fan-out: $8p + 12(p-1) \le 4096 \Rightarrow 20p \le 4108 \Rightarrow p = 205$.
3. A leaf holds (key, record pointer) pairs plus one next-leaf pointer: $20k + 8 \le 4096 \Rightarrow k = 204$ entries per leaf.
4. The fewest accesses come from the shortest possible tree, so assume every node is packed full. Leaves needed for 10^6 records: $\lceil 10^6/204 \rceil = 4902$.
5. Nodes reachable by level: root $= 1$, level 2 $= 205$, level 3 $= 205^2 = 42025$.
6. With two levels (root and leaves) there are at most 205 leaves, which is less than 4902. With three levels the bottom level can hold up to 42025 leaves, and $4902 \le 42025$. So the shortest tree has three levels: root, one internal level, leaves.
7. Nothing is in main memory at the start, so reading the root, one internal node and one leaf costs 3 disk accesses.
8. The leaf gives the address of the record's block. Each record fits in one block, so fetching it costs 1 more access.
9. Total minimum accesses $= 3 + 1 = 4$.

KEY IDEA
Work out the fan-out of a B+ tree node from the block size, find the fewest levels that can index all the records, then add one access for the record's own block.

WATCH OUT FOR
Forgetting the final access to the data block (answering 3), or assuming the root is already in memory when the question says no node is.

Q.55 NAT · 2 MARKS

Computer Networks › TCP

Consider a TCP connection between a client and a server with the following specifications: the round trip time is 6 ms, the size of the receiver advertised window is 50 KB, slow-start threshold at the client is 32 KB, and the maximum segment size is 2 KB. The connection is established at time t = 0. Assume that there are no timeouts and errors during transmission. Then the size of the congestion window (in KB) at time t + 60 ms after all acknowledgements are processed is _____.

Official answer: **44**

SOLUTION

1. Elapsed time is $60\ \text{ms}$ and $\text{RTT} = 6\ \text{ms}$, so $60/6 = 10$ round trips have completed, with all acknowledgements processed.
2. At $t = 0$ the connection starts in slow start with $\text{cwnd} = 1\ \text{MSS} = 2\ \text{KB}$.
3. In slow start the window doubles every RTT until it reaches the threshold of 32 KB:

After RTT	cwnd (KB)
0	2
1	4
2	8
3	16
4	32

4. After RTT 4, $\text{cwnd} = 32\ \text{KB} = \text{ssthresh}$, so slow start ends and congestion avoidance begins (additive increase).
5. In congestion avoidance the window grows by 1 MSS $= 2\ \text{KB}$ per RTT:

After RTT	cwnd (KB)
5	34
6	36
7	38
8	40
9	42
10	44

6. The receiver advertised window is 50 KB and $44 < 50$, so the sending window is not capped by the receiver. The congestion window after 10 RTTs is 44 KB.

KEY IDEA

Count RTTs (60/6 = 10). Slow start doubles cwnd each RTT from 2 KB up to ssthresh = 32 KB (4 RTTs), then congestion avoidance adds 1 MSS = 2 KB per RTT for the remaining 6 RTTs: 32 + 12 = 44 KB, below the 50 KB receiver window.

WATCH OUT FOR

Starting slow start from 1 KB instead of 1 MSS = 2 KB, or continuing to double past the threshold, or forgetting that the window growth switches to +1 MSS per RTT at ssthresh.

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 2020 question paper, organised by IIT Delhi; 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 Delhi 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