IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE CS · 2024 · SET 2

IAE Academy free study material · Updated 27 October 2026

GATE CS 2024 Set 2 Solved Paper

INSIDE

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

Question paper and official answer key: IISc Bengaluru

Online, with the questions linked to their topic notes: **iaeacademy.in/study/papers/cs-2024-2**

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: 32 MCQ · 18 MSQ · 15 NAT.

Official answer key: IISc Bengaluru · https://gate2024.iisc.ac.in/wp-content/uploads/2024/CS2FinalAnswerKey.pdf

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

This paper online, with the questions linked to their topic notes:
iaeacademy.in/study/papers/cs-2024-2

Answer key

Q	TYPE	MARKS	ANSWER
Q.1	MCQ	1	**C**
Q.2	MCQ	1	**A**
Q.3	MCQ	1	**A**
Q.4	MCQ	1	**C**
Q.5	MCQ	1	**B**
Q.6	MCQ	2	**C**
Q.7	MCQ	2	**C**
Q.8	MCQ	2	**D**
Q.9	MCQ	2	**A**
Q.10	MCQ	2	**A**
Q.11	MCQ	1	**C**
Q.12	MCQ	1	**A**
Q.13	MCQ	1	**A**
Q.14	MCQ	1	**B**
Q.15	MCQ	1	**A**
Q.16	MCQ	1	**B**
Q.17	MCQ	1	**B**
Q.18	MCQ	1	**B**
Q.19	MCQ	1	**A**
Q.20	MCQ	1	**A**
Q.21	MCQ	1	**B**
Q.22	MCQ	1	**A**
Q.23	MSQ	1	**B**
Q.24	MSQ	1	**B, C, D**
Q.25	MSQ	1	**A, B**
Q.26	MSQ	1	**A, B**
Q.27	MSQ	1	**A, C, D**
Q.28	MSQ	1	**B, C**
Q.29	MSQ	1	**B, D**
Q.30	MSQ	1	**B, C**
Q.31	MSQ	1	**A**
Q.32	MSQ	1	**C, D**
Q.33	MSQ	1	**A, B, D**
Q.34	NAT	1	**5**
Q.35	NAT	1	**3**
Q.36	MCQ	2	**B**
Q.37	MCQ	2	**B**
Q.38	MCQ	2	**A**
Q.39	MCQ	2	**A**
Q.40	MCQ	2	**A**
Q.41	MCQ	2	**B**
Q.42	MCQ	2	**B**
Q.43	MCQ	2	**A**
Q.44	MCQ	2	**D**
Q.45	MCQ	2	**B**
Q.46	MSQ	2	**B, C**
Q.47	MSQ	2	**A, B**
Q.48	MSQ	2	**B, C, D**
Q.49	MSQ	2	**A, B, D**
Q.50	MSQ	2	**C, D**
Q.51	MSQ	2	**D**
Q.52	MSQ	2	**B, C**
Q.53	NAT	2	**29.50 to 30.50**
Q.54	NAT	2	**4**
Q.55	NAT	2	**500**
Q.56	NAT	2	**50**
Q.57	NAT	2	**32**
Q.58	NAT	2	**2.9 to 3.1**
Q.59	NAT	2	**9**
Q.60	NAT	2	**2**
Q.61	NAT	2	**34**
Q.62	NAT	2	**15**
Q.63	NAT	2	**4**
Q.64	NAT	2	**1028**
Q.65	NAT	2	**9**

General Aptitude Q.1 to Q.10

Q.1 MCQ · 1 MARK

General Aptitude › Grammar & vocabulary

If '→' denotes increasing order of intensity, then the meaning of the words [walk → jog → sprint] is analogous to [bothered → _____ → daunted].

Which one of the given options is appropriate to fill the blank?

(A) phased

(B) phrased

(C) fazed

(D) fused

Official answer: **C**

SOLUTION

1. The arrow means increasing intensity: walk, jog and sprint are three levels of the same action, each stronger than the one before.
2. So the blank must be a word of the same kind as 'bothered' and 'daunted' (a state of mental disturbance), with an intensity between the two.
3. 'Bothered' is mild annoyance or worry. 'Daunted' is strong discouragement or intimidation. We need a middle state of disturbance.
4. 'Fazed' means disturbed, disconcerted or thrown off balance. That is stronger than 'bothered' and weaker than 'daunted'.
5. Check the others. 'Phased' (done in stages), 'phrased' (worded) and 'fused' (joined or melted together) carry no sense of mental disturbance, so they break the analogy. 'Phased' and 'phrased' also only sound like 'fazed'.
6. The sequence bothered → fazed → daunted rises in intensity, matching walk → jog → sprint, so the answer is (C).

KEY IDEA

In an intensity analogy, find the word of the same kind as the end words (a state of mental disturbance) whose strength lies between them: bothered, then fazed, then daunted.

WATCH OUT FOR

Picking 'phased' or 'phrased' because they sound like 'fazed'. These are homophones with unrelated meanings (done in stages; worded) and have no sense of disturbance, so they cannot fit the sequence.

Q.2 MCQ · 1 MARK

General Aptitude › Numerical

Two wizards try to create a spell using all the four elements, water, air, fire, and earth. For this, they decide to mix all these elements in all possible orders. They also decide to work independently. After trying all possible combination of elements, they conclude that the spell does not work.

How many attempts does each wizard make before coming to this conclusion, independently?

(A) 24

(B) 48

(C) 16

(D) 12

Official answer: **A**

SOLUTION

1. Each wizard mixes all four distinct elements (water, air, fire, earth) in all possible orders, so one attempt is one ordering of the four elements.
2. The number of arrangements of n distinct objects in a row is $n!$.
3. For $n = 4$: $4! = 4 \times 3 \times 2 \times 1 = 24$.
4. The wizards work independently, so each one separately goes through all 24 orderings. The number of wizards does not multiply the count for one wizard.
5. Each wizard makes 24 attempts, which is option (A).

KEY IDEA
Trying all possible orders of 4 distinct items is counting permutations, $4! = 24$. Independent working means each wizard repeats the same full count, so the counts are not added or multiplied.

WATCH OUT FOR
Multiplying by the number of wizards to get $2 \times 24 = 48$, even though the question asks for the attempts of each wizard.

Q.3 MCQ · 1 MARK

General Aptitude › Numerical

In an engineering college of 10,000 students, 1,500 like neither their core branches nor other branches. The number of students who like their core branches is 1/4th of the number of students who like other branches. The number of students who like both their core and other branches is 500.

The number of students who like their core branches is

(A) 1,800
(B) 3,500
(C) 1,600
(D) 1,500

Official answer: **A**

SOLUTION

1. Let C be the number of students who like their core branches and O the number who like other branches. Both counts include the 500 students who like both.
2. Students who like at least one kind of branch: $10{,}000 - 1{,}500 = 8{,}500$.
3. By the two-set union rule, $|C \cup O| = |C| + |O| - |C \cap O|$, so $8500 = C + O - 500$.
4. Hence $C + O = 9000$.
5. The given ratio is $C = \frac{1}{4}O$, so $O = 4C$.
6. Substituting, $C + 4C = 9000$, so $5C = 9000$ and $C = 1800$.
7. Check: $O = 7200$, and $1800 + 7200 - 500 = 8500$, which matches the number who like at least one kind of branch. So the answer is option (A).

KEY IDEA
Remove the students who like neither kind of branch, then apply the union rule $|C \cup O| = |C| + |O| - |C \cap O|$. With the ratio $O = 4C$ this gives one linear equation in C.

WATCH OUT FOR
Treating the 500 who like both as a separate group and not subtracting them once in the union rule, or forgetting to remove the 1,500 who like neither. Using $C + O = 8500$ gives $C = 1700$, which is not an option.

Q.4 MCQ · 1 MARK

General Aptitude › Numerical

For positive non-zero real variables x and y, if

$$\ln\left(\frac{x+y}{2}\right) = \frac{1}{2}\left[\ln(x) + \ln(y)\right]$$

then, the value of $\frac{x}{y} + \frac{y}{x}$ is

(A) 1

(B) 1/2

(C) 2

(D) 4

Official answer: **C**

SOLUTION

1. Combine the right side with the log rule: $\frac{1}{2}[\ln x + \ln y] = \frac{1}{2}\ln(xy) = \ln\sqrt{xy}$.
2. Both sides are logarithms of positive numbers and $\ln$ is one-to-one, so $\frac{x+y}{2} = \sqrt{xy}$.
3. Square both sides: $\frac{(x+y)^2}{4} = xy$, so $(x+y)^2 = 4xy$.
4. Expand: $x^2 + 2xy + y^2 = 4xy$, so $x^2 - 2xy + y^2 = 0$, that is $(x-y)^2 = 0$.
5. Hence $x = y$. This is the equality case of AM $\geq$ GM.
6. Then $\frac{x}{y} + \frac{y}{x} = 1 + 1 = 2$, which is option (C).

KEY IDEA

Turn the log equation into the statement that the arithmetic mean equals the geometric mean. For positive numbers this happens only when $x = y$.

WATCH OUT FOR

Writing $\ln\left(\frac{x+y}{2}\right) = \frac{1}{2}\left[\ln x + \ln y\right]$ and then splitting the left side as $\ln(x+y) - \ln 2$ without a plan, or treating $\ln(x+y)$ as $\ln x + \ln y$. Another slip is stopping at $\frac{x+y}{2} = \sqrt{xy}$ and not seeing that it forces $x = y$.

Q.5 MCQ · 1 MARK

General Aptitude › Numerical

In the sequence 6, 9, 14, x, 30, 41, a possible value of x is

(A) 25

(B) 21

(C) 18

(D) 20

Official answer: **B**

SOLUTION

1. Find the gaps between the known consecutive terms: $9 - 6 = 3$ and $14 - 9 = 5$.
2. These are consecutive odd numbers, so the gaps should continue $7, 9, 11$.
3. Then $x = 14 + 7 = 21$.
4. Check the terms after x: $30 - 21 = 9$ and $41 - 30 = 11$. Both gaps match the pattern, so the whole sequence $6, 9, 14, 21, 30, 41$ is consistent.
5. The same sequence is given by $n^2 + 5$ for $n = 1, 2, \ldots, 6$: $6, 9, 14, 21, 30, 41$.
6. Of the options, only 21 makes the sequence follow this rule. 25, 18 and 20 would break the pattern of gaps $3, 5, 7, 9, 11$. The answer is option (B).

KEY IDEA
Look at the differences between consecutive terms. They are consecutive odd numbers 3, 5, 7, 9, 11, which gives the rule $n^2 + 5$ and fixes $x = 21$.

WATCH OUT FOR
Stopping after the first two gaps, or assuming a constant gap or ratio, instead of testing the pattern against the terms after the blank (30 and 41).

Q.6 MCQ · 2 MARKS

General Aptitude › Reading

Sequence the following sentences in a coherent passage.

P: This fortuitous geological event generated a colossal amount of energy and heat that resulted in the rocks rising to an average height of 4 km across the contact zone.
Q: Thus, the geophysicists tend to think of the Himalayas as an active geological event rather than as a static geological feature.
R: The natural process of the cooling of this massive edifice absorbed large quantities of atmospheric carbon dioxide, altering the earth's atmosphere and making it better suited for life.
S: Many millennia ago, a breakaway chunk of bedrock from the Antarctic Plate collided with the massive Eurasian Plate.

(A) QPSR
(B) QSPR
(C) SPRQ
(D) SRPQ

Official answer: **C**

SOLUTION

1. Find the opening sentence. S introduces the event (a chunk of the Antarctic Plate colliding with the Eurasian Plate) and refers to nothing earlier, so it can open the passage. Q begins with "Thus", which draws a conclusion from earlier sentences. P begins with "This fortuitous geological event", which refers back to an event already described. Neither Q nor P can open the passage, so options A and B (both start with Q) are out.
2. Link P to S. "This fortuitous geological event" in P refers to the collision described in S, so P follows S directly.
3. Link R to P. "This massive edifice" in R refers to the mountain mass formed when the rocks rose, as described in P. R then describes how that mass cooled, so R follows P.
4. Place Q last. "Thus" marks a conclusion drawn from everything before it, namely that the Himalayas are an active geological event and not a static feature.
5. Test option D (SRPQ). R would follow S directly, but "this massive edifice" would have no antecedent, because the rising rocks appear only in P. So D fails.
6. The coherent order is S, P, R, Q, which is option C.

KEY IDEA
In sentence-ordering questions, find the sentence that needs no earlier context to open. Then chain the rest through back-references such as "this event" and "this massive edifice", and put "Thus" sentences at the end.

WATCH OUT FOR
Opening with Q because the Himalayas are the subject of the passage. Q begins with "Thus", so it is a conclusion and cannot open. Another slip is putting R straight after S, which leaves "this massive edifice" with nothing to refer to.

Q.7 MCQ · 2 MARKS

General Aptitude › Numerical

A person sold two different items at the same price. He made 10% profit in one item, and 10% loss in the other item. In selling these two items, the person made a total of

(A) 1% profit

(B) 2% profit

(C) 1% loss

(D) 2% loss

Official answer: **C**

SOLUTION

1. Let each item be sold at the same price S. Profit and loss percentages are taken on cost price, so first find each cost price.
2. Item sold at a 10% profit: $\mathrm{CP}_1 = \dfrac{S}{1.1} = \dfrac{10S}{11}$.
3. Item sold at a 10% loss: $\mathrm{CP}_2 = \dfrac{S}{0.9} = \dfrac{10S}{9}$.
4. Total cost price: $\dfrac{10S}{11} + \dfrac{10S}{9} = 10S \cdot \dfrac{9+11}{99} = \dfrac{200S}{99}$.
5. Total selling price: $2S = \dfrac{198S}{99}$. This is less than the total cost, so the overall result is a loss.
6. Loss $= \dfrac{200S}{99} - \dfrac{198S}{99} = \dfrac{2S}{99}$.
7. Loss percentage on the total cost $= \dfrac{2S/99}{200S/99} \times 100 = 1\%$.
8. So the person makes a 1% loss overall, which is option (C).

KEY IDEA

With equal selling prices, find each cost price by dividing the selling price by (1 + profit rate) or (1 - loss rate). Then compare the total cost with the total selling price. Percentages on different bases cannot be added or subtracted directly.

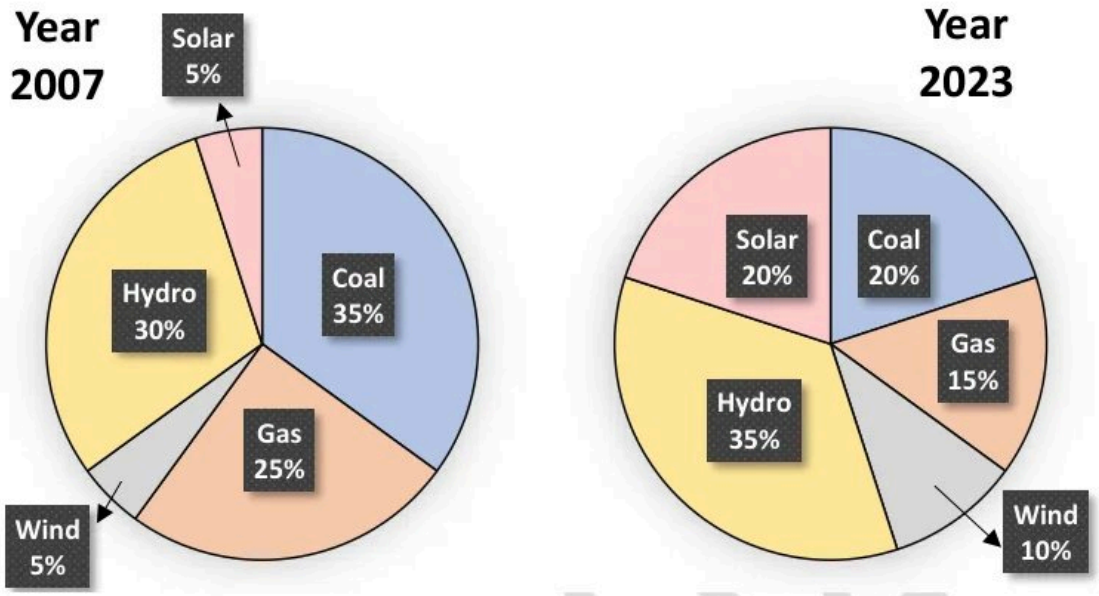
WATCH OUT FOR

Assuming that +10% and -10% cancel to 0%, or reading the 10% as a percentage of the selling price. Both profit and loss are on cost price, and the two cost prices differ. The loss item cost more, so the overall result is a loss.

Q.8 MCQ · 2 MARKS

General Aptitude › Data interpretation

Q.8 The pie charts depict the shares of various power generation technologies in the total electricity generation of a country for the years 2007 and 2023.



The renewable sources of electricity generation consist of Hydro, Solar and Wind. Assuming that the total electricity generated remains the same from 2007 to 2023, what is the percentage increase in the share of the renewable sources of electricity generation over this period?

(A) 25%

(B) 50%

(C) 77.5%

(D) 62.5%

Official answer: **D**

SOLUTION

1. Renewable sources are Hydro, Solar and Wind, so add their shares in each year.
2. Year 2007: $30\% + 5\% + 5\% = 40\%$ of the total (check: $35 + 25 + 5 + 30 + 5 = 100$).
3. Year 2023: $35\% + 20\% + 10\% = 65\%$ of the total (check: $20 + 15 + 10 + 35 + 20 = 100$).
4. The total generation is the same in both years, so the shares can be compared directly as amounts: the renewable output rose from 40 to 65 units per 100 units of total.
5. Percentage increase $= \dfrac{\text{new} - \text{old}}{\text{old}} \times 100 = \dfrac{65 - 40}{40} \times 100.$
6. $$\frac{25}{40} \times 100 = 62.5\%$$
7. This matches option (D).

KEY IDEA
Add the three renewable shares for each year. Because the total is unchanged, the percentage increase is (new share - old share) / old share x 100, taken relative to the old renewable share.

WATCH OUT FOR
Giving the difference in percentage points (65 - 40 = 25%, option A) instead of the relative increase over the 2007 renewable share. Another slip is dividing by 65 or by 100 instead of by 40.

Q.9 MCQ · 2 MARKS

General Aptitude › Spatial

A cube is to be cut into 8 pieces of equal size and shape. Here, each cut should be straight and it should not stop till it reaches the other end of the cube.

The minimum number of such cuts required is

(A) 3
(B) 4
(C) 7
(D) 8

Official answer: **A**

SOLUTION

1. Each cut is straight and goes right through to the other end, so it is a full plane cut. The pieces are not moved between cuts.
2. **Three cuts are enough.** Cut with a plane parallel to one pair of faces through the midpoints of the edges. This gives 2 equal halves. Cut again with a plane parallel to another pair of faces through the centre. This gives 4 equal pieces. Cut a third time with a plane parallel to the last pair of faces through the centre. This gives $2 \times 2 \times 2 = 8$ identical small cubes, each of equal size and shape.
3. **Two cuts are not enough.** Every piece of a cube is a convex solid, and one plane cuts a convex solid into at most 2 parts. The first cut gives at most 2 pieces. The second cut is one plane, so it splits each of those 2 pieces into at most 2 parts. That gives at most $2 \times 2 = 4$ pieces, which is fewer than 8. One cut gives at most 2 pieces, so it is also not enough.
4. So at least 3 cuts are needed, and 3 cuts do the job. The minimum number of cuts is 3, which is option (A).

KEY IDEA
Three mutually perpendicular plane cuts through the centre split a cube into $2 \times 2 \times 2 = 8$ equal cubes. Two cuts give at most 4 pieces, because each plane splits a convex piece into at most 2 parts.

WATCH OUT FOR
Picking 7 or 8 by counting one cut per extra piece, or picking 4 by cutting the cube into halves again and again. Each full cut can split every existing piece, so the number of pieces can double with each cut.

Q.10 MCQ · 2 MARKS

General Aptitude › Reasoning

Q.10 In the 4×4 array shown below, each cell of the first three rows has either a cross (X) or a number.

1	X	4	3
X	5	5	4
3	X	6	X

The number in a cell represents the count of the immediate neighboring cells (left, right, top, bottom, diagonals) NOT having a cross (X). Given that the last row has no crosses (X), the sum of the four numbers to be filled in the last row is

(A) 11

(B) 10

(C) 12

(D) 9

Official answer: **A**

SOLUTION

1. Write the grid by (row, column). Row 1: 1, X, 4, 3. Row 2: X, 5, 5, 4. Row 3: 3, X, 6, X. Row 4 is empty and has no crosses, so every row-4 neighbour counts as a cell without a cross.
2. Each number equals the number of neighbouring cells (left, right, up, down and the four diagonals) that are not crosses. Only row 3 and row 4 cells neighbour a row-4 cell, and the only crosses among them are $(3,2)$ and $(3,4)$.
3. Cell $(4,1)$ has 3 neighbours: $(3,1)$, $(3,2)$, $(4,2)$. Only $(3,2)$ is a cross, so the number is $3-1=2$.
4. Cell $(4,2)$ has 5 neighbours: $(3,1)$, $(3,2)$, $(3,3)$, $(4,1)$, $(4,3)$. Only $(3,2)$ is a cross, so the number is $5-1=4$.
5. Cell $(4,3)$ has 5 neighbours: $(3,2)$, $(3,3)$, $(3,4)$, $(4,2)$, $(4,4)$. The crosses are $(3,2)$ and $(3,4)$, so the number is $5-2=3$.
6. Cell $(4,4)$ has 3 neighbours: $(3,3)$, $(3,4)$, $(4,3)$. Only $(3,4)$ is a cross, so the number is $3-1=2$.
7. The last row reads $2,4,3,2$, and the sum is $2+4+3+2=11$.
8. Check against the given numbers. Cell $(3,3)=6$ has 8 neighbours with crosses at $(3,2)$ and $(3,4)$, so $8-2=6$. Cell $(3,1)=3$ has 5 neighbours with crosses at $(2,1)$ and $(3,2)$, so $5-2=3$. Cell $(2,2)=5$ has 8 neighbours with crosses at $(1,2)$, $(2,1)$ and $(3,2)$, so $8-3=5$. All three agree, so the cross positions are read correctly.

KEY IDEA

A cell's number is its count of existing neighbours (edge cells have fewer than 8) minus the number of neighbours that are crosses. For the last row, only the crosses in row 3 matter.

WATCH OUT FOR

Counting 8 neighbours for every cell, even though edge and corner cells have only 5 or 3. Another slip is leaving out diagonal neighbours or counting a cross as a neighbour without a cross.

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

Q.11 MCQ · 1 MARK

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

Consider a computer with a 4 MHz processor. Its DMA controller can transfer 8 bytes in 1 cycle from a device to main memory through cycle stealing at regular intervals. Which one of the following is the data transfer rate (in bits per second) of the DMA controller if 1% of the processor cycles are used for DMA?

(A) 2,56,000

(B) 3,200

(C) 25,60,000

(D) 32,000

Official answer: **C**

SOLUTION

1. A 4 MHz processor completes 4×10^6 clock cycles every second.
2. In cycle stealing, the DMA controller takes some of these cycles to use the bus. Here it takes 1% of them: $0.01 \times 4 \times 10^6 = 4 \times 10^4 = 40{,}000$ cycles per second.
3. In each stolen cycle it transfers 8 bytes, which is $8 \times 8 = 64$ bits.
4. Data transfer rate $= 40{,}000 \ \frac{\text{cycles}}{\text{s}} \times 64 \ \frac{\text{bits}}{\text{cycle}} = 2{,}560{,}000 \ \text{bits/s}$.
5. The options use Indian digit grouping, where $2{,}560{,}000$ is written 25,60,000. That is option (C). Option (A), 2,56,000, is ten times smaller and would need only 4,000 stolen cycles per second (0.1%).

KEY IDEA

With cycle stealing, DMA throughput is (cycles stolen per second) × (data moved per cycle). Cycles stolen per second = clock rate × the fraction of cycles given to DMA.

WATCH OUT FOR

The usual mistakes are misreading the Indian digit grouping (25,60,000 is 2.56 million and 2,56,000 is 256 thousand) and forgetting to turn bytes into bits. Leaving the answer in bytes gives 3,20,000 bytes/s, which is not an option. Taking 0.1% instead of 1% leads to option (A).

Q.12 MCQ · 1 MARK

Discrete Mathematics › Mathematical logic

Let p and q be the following propositions:

p: Fail grade can be given.

q: Student scores more than 50% marks.

Consider the statement: "Fail grade cannot be given when student scores more than 50% marks."

Which one of the following is the CORRECT representation of the above statement in propositional logic?

(A) $q \rightarrow \neg p$

(B) $q \rightarrow p$

(C) $p \rightarrow q$

(D) $\neg p \rightarrow q$

Official answer: **A**

SOLUTION

1. Read "X when Y" as "if Y, then X", so the clause after "when" is the hypothesis: $Y \rightarrow X$.
2. The hypothesis is "student scores more than 50% marks", which is q.

3. The conclusion is "fail grade cannot be given". This is the negation of p ("fail grade can be given"), so it is $\neg p$.
4. Combine them: $q \rightarrow \neg p$, which is option (A).
5. Check the other options. (B) $q \rightarrow p$ says a student scoring above 50% does get a fail grade, which is the opposite of the statement. (C) $p \rightarrow q$ says a fail grade implies a score above 50%, which reverses the direction and drops the negation. (D) $\neg p \rightarrow q$ says no fail grade implies a score above 50%, which is also not the statement. A counter-model for (C) and (D): take p true and q false, meaning a fail grade is given to a student at or below 50%. The original statement holds (q is false, so $q \rightarrow \neg p$ is true), but $p \rightarrow q$ is false. Take p false and q false: the original statement holds, but $\neg p \rightarrow q$ is false.
6. Equivalent forms of (A): $q \rightarrow \neg p \equiv \neg q \vee \neg p \equiv \neg(p \wedge q)$. A fail grade and a score above 50% never occur together.

KEY IDEA
"X when Y" means $Y \rightarrow X$: the clause after "when" is the hypothesis. "Cannot be given" negates the proposition p and puts $\neg p$ in the conclusion.

WATCH OUT FOR
Reading "fail grade cannot be given when student scores more than 50%" in the order the words appear, as $p \rightarrow q$ or $\neg p \rightarrow q$. Another slip is forgetting that "cannot" requires $\neg p$, which gives $q \rightarrow p$.

Q.13 MCQ · 1 MARK

Programming in C › Functions & scope

Consider the following C program. Assume parameters to a function are evaluated from *right to left*.

```
#include <stdio.h>

int g(int p) { printf("%d", p); return p; }
int h(int q) { printf("%d", q); return q; }

void f(int x, int y) {
  g(x);
  h(y);
}

int main() {
  f(g(10),h(20));
}
```

Which one of the following options is the CORRECT output of the above C program?

(A) 20101020
(B) 10202010
(C) 20102010
(D) 10201020

Official answer: **A**

SOLUTION

1. main() calls f(g(10), h(20)) . Since the arguments to f are evaluated right to left, h(20) is evaluated before g(10) .
2. h(20) runs first: printf("%d", q) with q = 20 prints 20 , and the call returns 20 .
3. g(10) runs next: printf("%d", p) with p = 10 prints 10 , and the call returns 10 .
4. Output so far (from evaluating the arguments): 2010 .

5. Now `f(x, y)` executes with its parameters bound by position: `x` receives the return value of `g(10)`, i.e. `x = 10`; `y` receives the return value of `h(20)`, i.e. `y = 20`. The order in which the arguments were evaluated does not change which parameter each one is bound to.
6. Inside `f`: the statement `g(x)` calls `g(10)`, printing `10` (its return value is discarded).
7. Next, `h(y)` calls `h(20)`, printing `20` (its return value is discarded).
8. Output added by the body of `f`: `1020`.
9. Concatenating everything printed, in order: `2010` + `1020` = `20101020`, which is option (A).

KEY IDEA

Separate two independent things: (1) the order in which argument *expressions* are evaluated (right to left here, which only affects the order of side-effect printing during the call `f(g(10), h(20))`), and (2) which value ends up bound to which *parameter* inside the called function (always by position: the first argument's value goes to the first parameter, regardless of evaluation order).

WATCH OUT FOR

Assuming that "right to left evaluation" also means the parameters are somehow bound out of order inside `f`, e.g. treating `y` as if it held `g(10)`'s value because `h(20)` was evaluated first. Evaluation order only changes when the side effects (the `printf` calls inside `g` and `h` as arguments) happen; it never changes that the first written argument's value goes to the first parameter (`x`) and the second to the second (`y`).

Q.14 MCQ · 1 MARK

Digital Electronics › Number systems

Q.14 The format of a single-precision floating-point number as per the **IEEE 754** standard is:

Sign (1bit)	Exponent (8 bits)	Mantissa (23 bits)

Choose the largest floating-point number among the following options.

(A)

Sign	Exponent	Mantissa
0	0111 1111	1111 1111 1111 1111 1111 111

(B)

Sign	Exponent	Mantissa
0	1111 1110	1111 1111 1111 1111 1111 111

(C)

Sign	Exponent	Mantissa
0	1111 1111	1111 1111 1111 1111 1111 111

(D)

Sign	Exponent	Mantissa
0	0111 1111	0000 0000 0000 0000 0000 000

Official answer: **B**

SOLUTION

1. All four options have sign bit 0, so all are non-negative and can be compared by exponent field first, then by mantissa.
2. For IEEE 754 single precision, a normal number (exponent field $1 \le e \le 254$) has value $(-1)^s \times 1.f \times 2^{e-127}$.
3. Option (C) has exponent field $1111\,1111 = 255$, which is reserved. With a non-zero mantissa (here all ones) it encodes NaN, which is not a number, so it cannot be the largest. (C) is ruled out.
4. Option (A) has $e = 0111\,1111 = 127$, so its value is $1.11\ldots1_2 \times 2^0 = 2 - 2^{-23} \approx 2$.
5. Option (D) has $e = 127$ and mantissa 0, so its value is $1.0 \times 2^0 = 1$.
6. Option (B) has $e = 1111\,1110 = 254$, so its value is $(2 - 2^{-23}) \times 2^{254-127} = (2 - 2^{-23}) \times 2^{127} \approx 3.4 \times 10^{38}$. This is the largest finite single-precision number.
7. Order of the valid numbers: (D) $<$ (A) $<$ (B). So the largest floating-point number is (B).

KEY IDEA
In IEEE 754 single precision, the exponent field 1111 1111 is reserved for infinity and NaN. The largest finite number therefore uses exponent 1111 1110 (254) with an all-ones mantissa.

WATCH OUT FOR
Picking (C) because it has the largest-looking bit pattern. An all-ones exponent with a non-zero mantissa is NaN, not a number.

Q.15 MCQ · 1 MARK

Data Structures and Algorithms › Complexity

Let $T(n)$ be the recurrence relation defined as follows:

$$T(0) = 1,$$
$$T(1) = 2, \text{ and}$$
$$T(n) = 5T(n-1) - 6T(n-2) \quad \text{for } n \geq 2$$

Which one of the following statements is TRUE?

(A) $T(n) = \Theta(2^n)$
(B) $T(n) = \Theta(n2^n)$
(C) $T(n) = \Theta(3^n)$
(D) $T(n) = \Theta(n3^n)$

Official answer: **A**

SOLUTION

1. The recurrence T(n) = 5T(n-1) - 6T(n-2) for n ≥ 2 is linear, homogeneous, with constant coefficients, so its closed form is determined by the characteristic equation.
2. Characteristic equation: $x^2 - 5x + 6 = 0$, which factors as $(x-2)(x-3) = 0$, giving roots $x = 2$ and $x = 3$.
3. General solution: $T(n) = A \cdot 2^n + B \cdot 3^n$ for constants A, B determined by the initial conditions.
4. Apply $T(0) = 1$: $A + B = 1$.
5. Apply $T(1) = 2$: $2A + 3B = 2$.
6. Subtract $2(A + B) = 2$ from $2A + 3B = 2$: $(2A + 3B) - (2A + 2B) = 2 - 2 \Rightarrow B = 0$, so $A = 1$.
7. Thus $T(n) = 2^n$ exactly. Verification: $T(2) = 5(2) - 6(1) = 4 = 2^2$ and $T(3) = 5(4) - 6(2) = 8 = 2^3$, both consistent with the closed form.
8. Since $T(n) = 2^n$ exactly for all n, $T(n) = \Theta(2^n)$, which is option A.

KEY IDEA
A linear homogeneous recurrence with constant coefficients is solved via its characteristic equation; the roots give the exact closed form, and initial conditions pin down the coefficients.

WATCH OUT FOR
Assuming the larger root (3) always dominates the asymptotic growth without checking whether its coefficient in the closed form is actually zero: here the initial conditions force the coefficient of 3^n to vanish, so the recurrence in fact grows only as 2^n, not 3^n.

Q.16 MCQ · 1 MARK

Engineering Mathematics › Calculus

Let $f(x)$ be a continuous function from $\mathbb{R}$ to $\mathbb{R}$ such that

$$f(x) = 1 - f(2 - x)$$

Which one of the following options is the CORRECT value of $\int_0^2 f(x)\,dx$?

(A) 0
(B) 1

(C) 2

(D) -1

Official answer: **B**

SOLUTION

1. Let $I = \int_0^2 f(x)\,dx$.
2. Substitute the given relation $f(x) = 1 - f(2-x)$ inside the integral:
$$I = \int_0^2 \big(1 - f(2-x)\big)\,dx = 2 - \int_0^2 f(2-x)\,dx.$$
3. For any continuous g, the substitution $u = a - x$ gives $\int_0^a g(x)\,dx = \int_0^a g(a-x)\,dx$. With $a = 2$ and $g = f$, $\int_0^2 f(2-x)\,dx = \int_0^2 f(x)\,dx = I$.
4. Hence $I = 2 - I$, so $2I = 2$ and $I = 1$.
5. Check with $f(x) = \frac{1}{2}$, which satisfies $f(x) = 1 - f(2-x)$: $\int_0^2 \frac{1}{2}\,dx = 1$. The answer is option (B).

KEY IDEA

The relation $f(x) + f(2-x) = 1$ pairs each point with its mirror about $x = 1$. The integral of $f(2-x)$ over $[0, 2]$ equals the integral of $f(x)$, so $2I = \int_0^2 1\,dx = 2$.

WATCH OUT FOR

Assuming $\int_0^2 f(2-x)\,dx = -\int_0^2 f(x)\,dx$ because of the sign flip in the argument. The limits flip with the substitution, which cancels the sign, so the two integrals are equal. That mistake gives $I = 2 + I$, which has no solution.

Q.17 MCQ · 1 MARK

Discrete Mathematics › Graph theory

Let A be the adjacency matrix of a simple undirected graph G. Suppose A is its own inverse. Which one of the following statements is always TRUE?

(A) G is a cycle

(B) G is a perfect matching

(C) G is a complete graph

(D) There is no such graph G

Official answer: **B**

SOLUTION

1. A is its own inverse, so $A^2 = I$.
2. For a simple undirected graph, A is symmetric with zero diagonal. Then $(A^2)_{ij} = \sum_k A_{ik}A_{kj}$ is the number of common neighbours of i and j. For $i = j$ it equals $\deg(i)$.
3. Diagonal entries: $(A^2)_{ii} = \deg(i) = 1$, so every vertex has degree exactly 1.
4. Off-diagonal entries: $(A^2)_{ij} = 0$ for $i \neq j$. This holds automatically when every degree is 1, because two distinct vertices cannot share a neighbour (that neighbour would have degree at least 2).
5. A graph in which every vertex has degree 1 is a set of disjoint edges, that is, a perfect matching.
6. Such graphs exist. For K_2, $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $A^2 = I$, so (D) is false.
7. A cycle has every degree equal to 2, and K_n for $n \geq 3$ has every degree equal to $n - 1 \geq 2$. Both violate degree 1, so (A) and (C) are not true. Only (B) always holds.

KEY IDEA
$A^2 = I$ forces every vertex degree to be 1 (the diagonal of A^2 is the degree). A graph with all degrees 1 is a perfect matching, and the off-diagonal zeros then hold automatically.

WATCH OUT FOR
Choosing (D) because "an inverse adjacency matrix looks impossible". Check K_2, whose adjacency matrix is its own inverse. Another slip is forgetting that $(A^2)_{ii}$ is the degree, not 0 or 1 by default.

Q.18 MCQ · 1 MARK

Engineering Mathematics › Probability & statistics

When six unbiased dice are rolled simultaneously, the probability of getting all distinct numbers (i.e., 1, 2, 3, 4, 5, and 6) is

(A) $\frac{1}{324}$
(B) $\frac{5}{324}$
(C) $\frac{7}{324}$
(D) $\frac{11}{324}$

Official answer: **B**

SOLUTION

1. Each die has 6 faces, so rolling six dice gives $6^6 = 46656$ equally likely ordered outcomes (multiplication rule).
2. Favourable outcomes: the six dice show all of $1, 2, \ldots, 6$, each exactly once. Each such outcome is an arrangement of the six numbers over the six dice, so the count is $6! = 720$.
3. $$P = \frac{6!}{6^6} = \frac{720}{46656}$$
4. Factorise: $720 = 2^4 \cdot 3^2 \cdot 5$ and $46656 = 2^6 \cdot 3^6$. Cancelling $2^4 \cdot 3^2$ gives $$P = \frac{5}{2^2 \cdot 3^4} = \frac{5}{324}.$$
5. This is option (B).

KEY IDEA
Count favourable outcomes as ordered arrangements: all six faces distinct means the dice show a permutation of 1 to 6, giving $6!$ out of 6^6 equally likely outcomes.

WATCH OUT FOR
Using $\binom{6}{6}$ or $6!/6$ instead of $6!$, or forgetting to treat the dice as distinguishable. Both the numerator and the denominator must count ordered outcomes.

Q.19 MCQ · 1 MARK

Database Management and Warehousing › Transactions

Once the DBMS informs the user that a transaction has been successfully completed, its effect should persist even if the system crashes before all its changes are reflected on disk. This property is called

(A) durability
(B) atomicity
(C) consistency
(D) isolation

Official answer: **A**

SOLUTION

1. The ACID properties are atomicity, consistency, isolation and durability. Match the described behaviour to one of them.
2. The question describes a transaction that the DBMS has reported as successfully completed (committed). Its effect must persist even if the system crashes before all its changes have been written

to disk.

3. This is the guarantee that committed changes are permanent. The recovery manager provides it with a write-ahead log (WAL). The log records of a committed transaction are forced to stable storage before the commit is reported. After a crash, recovery REDOes the committed transactions from the log.
4. Atomicity means all-or-nothing execution, so incomplete work is undone. Consistency means a transaction takes the database from one valid state to another. Isolation means concurrent transactions do not interfere with one another. None of these is about committed effects surviving a crash.
5. So the property is durability, option (A).

KEY IDEA
Durability means that once a transaction has committed, its effects persist even if the system crashes. Recovery from the write-ahead log (REDO of committed work) provides this.

WATCH OUT FOR
Choosing atomicity because a crash is involved. Atomicity concerns undoing a transaction that was incomplete at the crash. The question is about a transaction already reported as completed whose effects must still survive, and that is durability.

Q.20 MCQ · 1 MARK

Database Management and Warehousing › ER model

In the context of owner and weak entity sets in the ER (Entity-Relationship) data model, which one of the following statements is TRUE?

(A) The weak entity set MUST have total participation in the identifying relationship

(B) The owner entity set MUST have total participation in the identifying relationship

(C) Both weak and owner entity sets MUST have total participation in the identifying relationship

(D) Neither weak entity set nor owner entity set MUST have total participation in the identifying relationship

Official answer: **A**

SOLUTION

1. A weak entity set has no key of its own. Each weak entity is identified by its partial key (discriminator) together with the primary key of one owner entity.
2. The identifying relationship links the weak entity set to its owner entity set. It is many-to-one from weak to owner.
3. A weak entity with no owner could not be identified at all. So every weak entity must be related to some owner, which means the weak entity set has **total participation** in the identifying relationship.
4. An owner entity may have no dependents. For example, an employee may have no dependents, so the owner set need not participate totally.
5. So only statement (A) must hold. (B) and (C) wrongly require the owner to participate totally, and (D) denies the weak set's total participation.

KEY IDEA
A weak entity exists only through its owner, so every weak entity must take part in the identifying relationship, which is total participation on the weak side. The owner side is unconstrained.

WATCH OUT FOR
Thinking the owner must also participate totally. An owner can have no weak entities, such as an employee with no dependents.

Q.21 MCQ · 1 MARK

Compiler Design › Lexical analysis

Consider the following two sets:

Set X: P. Lexical Analyzer; Q. Syntax Analyzer; R. Intermediate Code Generator; S. Code Optimizer
Set Y: 1. Abstract Syntax Tree; 2. Token; 3. Parse Tree; 4. Constant Folding

Which one of the following options is the CORRECT match from Set X to Set Y ?

(A) P – 4; Q – 1; R – 3; S – 2

(B) P – 2; Q – 3; R – 1; S – 4

(C) P – 2; Q – 1; R – 3; S – 4

(D) P – 4; Q – 3; R – 2; S – 1

Official answer: **B**

SOLUTION

1. The lexical analyzer reads the character stream and groups it into lexemes, each reported as a **token**. So P matches 2 (Token).
2. The syntax analyzer (parser) checks the token stream against the grammar and builds the **parse tree**. So Q matches 3 (Parse Tree).
3. The intermediate code generator works from the **abstract syntax tree** that the front end passes on, a compact form of the parse tree. So R matches 1 (Abstract Syntax Tree).
4. The code optimizer applies transformations such as **constant folding**, which evaluates constant expressions at compile time. So S matches 4 (Constant Folding).
5. The matching is P – 2; Q – 3; R – 1; S – 4. Option (A) pairs P with 4 and option (C) pairs Q with 1. Option (D) pairs P with 4. Only option (B) has all four pairs right.

KEY IDEA

Each compiler phase has its own characteristic output or operation. The lexer produces tokens, the parser produces a parse tree, the intermediate code generator takes the syntax tree (AST), and the optimizer performs transformations such as constant folding.

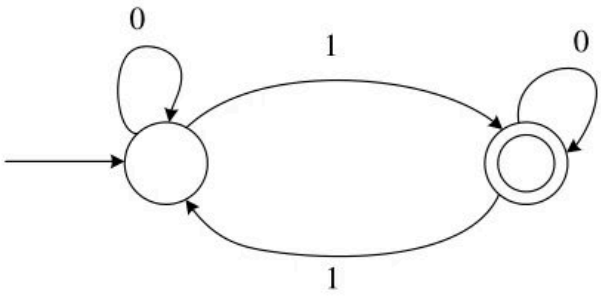
WATCH OUT FOR

Matching the parse tree to the intermediate code generator, or the abstract syntax tree to the parser, because the two trees look alike. The parser's own output is the parse tree, and the AST is the compact tree the later phases use.

Q.22 MCQ · 1 MARK

Theory of Computation › Finite automata

Q.22 Which one of the following regular expressions is equivalent to the language accepted by the DFA given below?



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

(B) $0^*(10^*11)^*0^*$

(C) $0^*1(010^*1)^*0^*$

(D) $0(1+0^*10^*1)^*0^*$

Official answer: **A**

SOLUTION

1. Read the DFA: the start state is q_0 (not accepting) and q_1 is the only accepting state. Both states have a self-loop on 0, and the symbol 1 moves $q_0 \to q_1$ and $q_1 \to q_0$. So 0 never changes the state and 1 toggles it. The language is the set of strings over $\{0,1\}$ with an odd number of 1s.
2. Derive the expression by state elimination. To go from q_0 to q_1 for the first time, read any number of 0s at q_0 and then a 1. This gives the prefix 0^*1.
3. Every way of leaving q_1 and returning to q_1 is one of two kinds. Either read a 0 (the self-loop), or read a 1 to q_0, then 0^* at q_0, then a 1 back to q_1, which is 10^*1. So the loop part is $(0+10^*1)^*$.
4. Therefore $L = 0^*1(0+10^*1)^*$, which is option (A).
5. Check the other options with short strings. The string 1 has one 1 and is accepted. (B) $0^*(10^*11)^*0^*$ cannot generate 1, since every 1 in it comes in a block of three 1s or none. (D) $0(1+0^*10^*1)^*0^*$ must begin with 0, so it cannot generate 1. (C) $0^*1(010^*1)^*0^*$ cannot generate 111, which has three 1s and is accepted. After the first 1, every block must start with 0, and the trailing part is only 0s. Also test (A) on 1011, which has three 1s and is accepted: $1 \cdot 0 \cdot (10^*1)$ with $10^*1 = 11$ gives 1011, so (A) generates it.

KEY IDEA

Each 0 leaves the state unchanged and each 1 toggles it, so the DFA accepts exactly the strings with an odd number of 1s. State elimination gives 0^*1 to reach the accepting state for the first time, followed by a loop of 0 or 10^*1.

WATCH OUT FOR

Forgetting that the loop at q_0 is 0^* inside the $1\ldots1$ block, which gives 10^*1 rather than 11 or 101. Another mistake is picking an expression that cannot produce the single string 1, or one that forces a 0 after the first 1 (as in option C), which wrongly rejects strings such as 111.

Q.23 MSQ · 1 MARK

Computer Networks › IPv4

Node X has a TCP connection open to node Y. The packets from X to Y go through an intermediate IP router R. Ethernet switch S is the first switch on the network path between X and R. Consider a packet sent from X to Y over this connection.

Which of the following statements is/are TRUE about the destination IP and MAC addresses on this packet at the time it leaves X?

(A) The destination IP address is the IP address of R

(B) The destination IP address is the IP address of Y

(C) The destination MAC address is the MAC address of S

(D) The destination MAC address is the MAC address of Y

Official answer: **B**

SOLUTION

1. The IP header carries end-to-end addresses. The packet is meant for Y, so the destination IP is the IP address of Y, and a router does not change it when forwarding. So (B) is true and (A) is false.
2. X and Y are on different networks, because the path between them includes the router R. X sees that Y is not in its own subnet, so it hands the packet to its default gateway, which is R.
3. An Ethernet frame is addressed to the next hop on the same link. X uses ARP to find the MAC address of R, so the destination MAC on the frame leaving X is the MAC address of R.
4. Switch S works at layer 2 and is transparent to the frame. It forwards the frame by looking up the destination MAC address, and the frame is not addressed to S. So (C) is false.
5. The MAC address of Y is not the destination MAC on the X to R link. It appears only on the last link, between the final router and Y. So (D) is false.
6. Only (B) is true.

KEY IDEA

The destination IP address stays the same from source to destination. The destination MAC address is rewritten at every hop and always names the next hop on the current link, which for X is the default gateway R.

WATCH OUT FOR

Choosing (A) by thinking the IP address also points to the next hop, or choosing (D) by thinking the MAC address is end-to-end. Another mistake is choosing (C), because a switch is only a pass-through at layer 2 and never the addressee of the frame.

Q.24 MSQ · 1 MARK

Operating Systems › Memory management

Which of the following tasks is/are the responsibility/responsibilities of the memory management unit (MMU) in a system with paging-based memory management?

(A) Allocate a new page table for a newly created process

(B) Translate a virtual address to a physical address using the page table

(C) Raise a trap when a virtual address is not found in the page table

(D) Raise a trap when a process tries to write to a page marked with read-only permission in the page table

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

SOLUTION

1. The MMU is the hardware between the CPU and memory. On every memory reference it does the work that must happen at hardware speed. Decisions about what to allocate belong to the operating system.
2. (A) Allocating a page table for a new process is done by the OS kernel when it creates the process. The MMU only reads the table, through the page-table base register. So (A) is not an MMU task.
3. (B) Translation is the core MMU function. It splits the virtual address into a page number and an offset, finds the frame (through the TLB or a page-table walk) and forms the physical address. So (B) is an MMU task.

4. (C) If the page-table entry for the page is absent or its valid bit is 0, the MMU detects this during translation and raises a page-fault trap. The OS then handles the fault. Detecting the problem and raising the trap are done by the MMU, so (C) is an MMU task.
5. (D) Each page-table entry carries protection bits (read, write, execute). The MMU checks them on every access. A write to a read-only page raises a protection-fault trap. So (D) is an MMU task.
6. Correct options: B, C and D.

KEY IDEA
The MMU is hardware. It translates addresses and checks the valid and permission bits in the page-table entry, trapping to the OS on a violation. The OS does all allocation and policy work.

WATCH OUT FOR
Counting the creation of a page table as an MMU task. The MMU only uses the table. The kernel builds it, and the kernel handles the trap the MMU raises.

Q.25 MSQ · 1 MARK

Operating Systems › Processes & threads

Consider a process P running on a CPU. Which one or more of the following events will always trigger a context switch by the OS that results in process P moving to a non-running state (e.g., ready, blocked)?

(A) P makes a blocking system call to read a block of data from the disk

(B) P tries to access a page that is in the swap space, triggering a page fault

(C) An interrupt is raised by the disk to deliver data requested by some other process

(D) A timer interrupt is raised by the hardware

Official answer: **A, B**

SOLUTION

1. The question asks which events will **always** make the OS switch context so that P leaves the running state. An event that only sometimes does this is not correct.
2. (A) A blocking system call to read a block from the disk cannot finish at once, because the disk transfer takes time. The OS moves P to the blocked state and dispatches another process, so a context switch always happens.
3. (B) The page is in swap space, so it is not in memory. The page fault handler must start a disk read to bring it in, and P cannot continue until that I/O completes. P moves to the blocked state and the CPU goes to another process, so a context switch always happens.
4. (C) The disk interrupt delivers data for some other process. The handler may move that process from blocked to ready, but P is still runnable. The scheduler need not preempt P, so P need not leave the CPU. Not guaranteed.
5. (D) A timer interrupt invokes the scheduler, but the scheduler may let P continue. For example, P may be the only ready process, or its time slice may not be used up. So a context switch is not guaranteed.
6. Only (A) and (B) force P out of the running state every time.

KEY IDEA
P always leaves the CPU when it must wait for I/O that it started itself, as with a blocking read or a page fault on a swapped-out page. Interrupts that merely invoke the scheduler or serve other processes leave the decision open.

WATCH OUT FOR
Treating a timer interrupt as always causing a context switch. It only gives the scheduler a chance to switch, and the scheduler may keep P running.

Q.26 MSQ · 1 MARK

Database Management and Warehousing › Indexing

Which of the following file organizations is/are I/O efficient for the scan operation in DBMS?

(A) Sorted

(B) Heap

(C) Unclustered tree index

(D) Unclustered hash index

Official answer: **A, B**

SOLUTION

1. A full scan reads every record of the file, so its I/O cost is the number of pages that hold the data. An organization is efficient for a scan when it stores the records densely and lets the pages be read one after another.
2. Heap file: records are packed into pages with no gaps. A scan costs about B page reads, where B is the number of data pages, and the reads are sequential. This is efficient.
3. Sorted file: the records are also stored densely, in key order, so a full scan again costs about B sequential page reads. The sort order does not hurt a scan. This is efficient.
4. Unclustered tree index: the leaf entries are in key order, but the data records are scattered across pages. Scanning in index order needs about one page read per record, so about N random reads, where N is the number of records. Since N is much larger than B, this is inefficient.
5. Unclustered hash index: the buckets are not in key order and each entry points to a record on an arbitrary page. A scan again costs about one page read per record, plus the index overhead. This is inefficient.
6. So only the sorted file (A) and the heap file (B) are I/O efficient for a scan.

KEY IDEA
A scan costs the number of data pages, so it is cheap only when the records are packed densely in the file itself (heap or sorted). An unclustered index leaves the records scattered, so a scan through it costs about one read per record.

WATCH OUT FOR
Thinking an index always helps. An unclustered index helps a search on one key, but it makes a full scan cost about one page read per record instead of B sequential page reads.

Q.27 MSQ · 1 MARK

Database Management and Warehousing › Transactions

Which of the following statements about the Two Phase Locking (2PL) protocol is/are TRUE?

(A) 2PL permits only serializable schedules

(B) With 2PL, a transaction always locks the data item being read or written just before every operation and always releases the lock just after the operation

(C) With 2PL, once a lock is released on any data item inside a transaction, no more locks on any data item can be obtained inside that transaction

(D) A deadlock is possible with 2PL

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

SOLUTION

1. Recall the rule: in Two Phase Locking every transaction has a growing phase (it only acquires locks) followed by a shrinking phase (it only releases locks). The first unlock marks the lock point, and no lock may be acquired after it.
2. (A) Every schedule allowed by 2PL is conflict serializable, because the transactions can be ordered by their lock points. So 2PL permits only serializable schedules. TRUE.
3. (B) Locking an item just before each operation and releasing it just after means the transaction unlocks one item and then locks the next. That is an unlock followed by a lock, which breaks the two-phase

rule. Basic 2PL does not require per-operation locking and releasing, and a transaction that did this would not be following 2PL. FALSE.

4. (C) This is the definition of 2PL: once any lock has been released (the shrinking phase has begun), no new lock on any data item can be obtained by that transaction. TRUE.
5. (D) 2PL does not prevent deadlock. For example, T_1 holds an exclusive lock on A and requests one on B, while T_2 holds an exclusive lock on B and requests one on A. Both are in their growing phase and wait for each other forever. TRUE.
6. The true statements are A, C and D.

KEY IDEA

Two Phase Locking means no lock is acquired after the first unlock. This guarantees conflict serializability, but it does not prevent deadlock.

WATCH OUT FOR

Thinking that 2PL is deadlock-free because it is serializable. Deadlock freedom is a separate property, so 2PL needs deadlock detection or prevention on top. Another slip is taking per-operation lock and release (B) as 2PL, when it is the opposite of two-phase behaviour.

Q.28 MSQ · 1 MARK

Computer Networks › IPv4

Which of the following statements about IPv4 fragmentation is/are TRUE?

(A) The fragmentation of an IP datagram is performed only at the source of the datagram

(B) The fragmentation of an IP datagram is performed at any IP router which finds that the size of the datagram to be transmitted exceeds the MTU

(C) The reassembly of fragments is performed only at the destination of the datagram

(D) The reassembly of fragments is performed at all intermediate routers along the path from the source to the destination

Official answer: **B, C**

SOLUTION

1. In IPv4, a router that must forward a datagram onto a link whose MTU is smaller than the datagram, and whose DF (Don't Fragment) bit is not set, fragments it. Fragmentation is therefore not limited to the source host. (A) is false.
2. Each link on the path can have a different MTU, so any router may find that a datagram is larger than the MTU of its outgoing link and fragment it. (B) is true.
3. Fragments travel as independent datagrams and may take different routes, so an intermediate router cannot be sure of seeing all fragments of a datagram. IPv4 therefore reassembles only at the final destination host. (C) is true.
4. Intermediate routers forward fragments like any other datagram and do not reassemble them. (D) is false.
5. The true statements are (B) and (C).

KEY IDEA

IPv4 fragments at any router whose outgoing link MTU is smaller than the datagram (if DF is not set), but reassembles only at the destination host, because fragments may take different paths.

WATCH OUT FOR

Thinking that only the source can fragment, as in IPv6, where routers never fragment. In IPv4, routers do fragment, but they never reassemble.

Q.29 MSQ · 1 MARK

Compiler Design › Syntax-directed translation

Which of the following statements is/are FALSE?

(A) An attribute grammar is a syntax-directed definition (SDD) in which the functions in the semantic rules have no side effects

(B) The attributes in a L-attributed definition cannot always be evaluated in a depth-first order

(C) Synthesized attributes can be evaluated by a bottom-up parser as the input is parsed

(D) All L-attributed definitions based on LR(1) grammar can be evaluated using a bottom-up parsing strategy

Official answer: **B, D**

SOLUTION

1. We need the FALSE statements, so test each option against the definitions.
2. (A) An attribute grammar is an SDD whose semantic rules are functions with no side effects. This is the standard definition, so (A) is TRUE.
3. (B) In an L-attributed definition, an inherited attribute of a symbol depends only on the inherited attributes of its parent and on attributes of symbols to its left. A left-to-right depth-first traversal of the parse tree therefore always finds every value it needs already computed. So the attributes can always be evaluated in depth-first order. The claim that they cannot always be is FALSE.
4. (C) A bottom-up parser can compute a synthesized attribute at each reduction from the attributes of the right-hand-side symbols already on the stack. This is how S-attributed definitions work, so (C) is TRUE.
5. (D) Evaluating an L-attributed definition bottom-up needs rewriting: each inherited attribute needs a marker nonterminal $M \to \varepsilon$ inserted before the symbol. This rewriting is guaranteed to work only for LL(1) grammars. For an LR(1) grammar the markers can introduce conflicts. Example: take $S \to a\,A\,c \mid a\,B\,d$, $A \to e$, $B \to e$, which is LR(1), and suppose A and B each have an inherited attribute. After inserting markers, $S \to a\,M_1\,A\,c \mid a\,M_2\,B\,d$ with $M_1 \to \varepsilon$ and $M_2 \to \varepsilon$. After shifting a with lookahead e, the parser must reduce by $M_1 \to \varepsilon$ or by $M_2 \to \varepsilon$ and cannot decide. This is a reduce-reduce conflict. So not all L-attributed definitions on LR(1) grammars can be evaluated bottom-up, and (D) is FALSE.
6. The false statements are (B) and (D).

KEY IDEA

L-attributed definitions can always be evaluated in one left-to-right depth-first pass. Evaluating them bottom-up is guaranteed only for LL(1) grammars, because the marker nonterminals that inherited attributes need can break an LR(1) grammar.

WATCH OUT FOR

Reading (B) as true because bottom-up parsing cannot evaluate inherited attributes directly. The question there is depth-first evaluation order, which L-attribution always allows. Another slip is assuming that the LL(1) result for bottom-up evaluation extends to every LR(1) grammar.

Q.30 MSQ · 1 MARK

Digital Electronics › Boolean logic

For a Boolean variable x, which of the following statements is/are FALSE?

(A) $x \cdot 1 = x$

(B) $x + 1 = x$

(C) $x \cdot x = 0$

(D) $x + \overline{x} = 1$

Official answer: **B, C**

SOLUTION

1. A statement is FALSE if it fails for at least one value of $x \in \{0, 1\}$, so check each option against both values.
2. (A) $x \cdot 1 = x$ (identity law). For $x = 0$: $0 \cdot 1 = 0$. For $x = 1$: $1 \cdot 1 = 1$. Both agree, so this is TRUE.
3. (B) $x + 1 = x$. By the null (domination) law $x + 1 = 1$ for every x. For $x = 0$: left side $0 + 1 = 1$, right side 0. They differ, so this is FALSE.
4. (C) $x \cdot x = 0$. By the idempotent law $x \cdot x = x$. For $x = 1$: left side $1 \cdot 1 = 1$, right side 0. They differ, so this is FALSE.
5. (D) $x + \overline{x} = 1$ (complement law). For $x = 0$: $0 + 1 = 1$. For $x = 1$: $1 + 0 = 1$. Both agree, so this is TRUE.
6. The FALSE statements are (B) and (C).

KEY IDEA

With a single Boolean variable there are only two cases, $x = 0$ and $x = 1$. A statement is false if it fails for either one, so a short truth-table check settles each option.

WATCH OUT FOR

Misreading the question: it asks which statements are FALSE, not which are true. Another slip is mixing up the null law $x + 1 = 1$ with the identity law $x + 0 = x$, or the idempotent law $x \cdot x = x$ with the complement law $x \cdot \overline{x} = 0$.

Q.31 MSQ · 1 MARK

Computer Organization and Architecture › Pipelining

An instruction format has the following structure:

Instruction Number: *Opcode destination reg, source reg-1, source reg-2*

Consider the following sequence of instructions to be executed in a pipelined processor:

```
I1: DIV R3, R1, R2
I2: SUB R5, R3, R4
I3: ADD R3, R5, R6
I4: MUL R7, R3, R8
```

Which of the following statements is/are TRUE?

(A) There is a RAW dependency on R3 between I1 and I2

(B) There is a WAR dependency on R3 between I1 and I3

(C) There is a RAW dependency on R3 between I2 and I3

(D) There is a WAW dependency on R3 between I3 and I4

Official answer: **A**

SOLUTION

1. List every access to R3 across the four instructions, in program order, marking each as a read or a write (destination = write, source = read):

```
I1: DIV R3, R1, R2   → writes R3
I2: SUB R5, R3, R4   → reads  R3 (source reg-1)
I3: ADD R3, R5, R6   → writes R3
I4: MUL R7, R3, R8   → reads  R3 (source reg-1)
```

2. **I1 → I2 (Statement A):** I1 writes R3 and I2, which comes later in program order, reads that same R3 as an operand. A later read of a value an earlier instruction produces is exactly a **RAW (read-after-**

write / true) dependency. So Statement A is TRUE.

3. **I1 → I3 (Statement B):** A WAR (write-after-read) dependency on a register requires an earlier instruction to *read* it and a later instruction to *write* it. I1 only writes R3 (it never reads R3), so the I1-I3 pair cannot be WAR on R3 (it is in fact WAW, since both write R3). Statement B is FALSE.
4. **I2 → I3 (Statement C):** I2 *reads* R3 (as SUB's source reg-1) and I3, later in program order, *writes* R3. A later write following an earlier read of the same register is the definition of a **WAR (anti-) dependency**, not RAW. Statement C mislabels this as RAW, so it is FALSE.
5. **I3 → I4 (Statement D):** A WAW dependency on a register requires both instructions to *write* it. I3 writes R3, but I4 only *reads* R3 (as MUL's source reg-1); I4's destination register is R7, not R3. Since I4 never writes R3, there is no WAW dependency on R3 between I3 and I4. Statement D is FALSE.
6. Only Statement A correctly names a true dependency on R3. Since this is an MSQ, the final answer is exactly the option(s) verified true: **A**.

KEY IDEA

To classify a data hazard between two instructions sharing a register, first tag every access to that register, in program order, as a read (source) or a write (destination): earlier-write/later-read = RAW (true), earlier-read/later-write = WAR (anti), and earlier-write/later-write = WAW (output). Only a register that is genuinely written by BOTH instructions can be WAW, and only one genuinely read-then-written can be WAR. Mislabeling one for the other is the entire trap in this question.

WATCH OUT FOR

Assuming that because two instructions both "touch" the same register in some order, any of RAW/WAR/WAW could apply interchangeably, e.g. calling the I2(reads R3)→I3(writes R3) pair a RAW dependency (as option C does) instead of correctly recognizing it as WAR, or calling I1(writes R3)→I3(writes R3) a WAR (as option B does) instead of WAW. The fix is to always ask two separate yes/no questions per pair: "does the earlier instruction read or write this register?" and "does the later instruction read or write it?", rather than pattern-matching on which instructions appear first.

Q.32 MSQ · 1 MARK

Computer Networks › IPv4

Which of the following fields of an IP header is/are *always* modified by any router before it forwards the IP packet?

(A) Source IP Address

(B) Protocol

(C) Time to Live (TTL)

(D) Header Checksum

Official answer: **C, D**

SOLUTION

1. A router that forwards an IPv4 packet must decrement the Time to Live (TTL) field by 1, so that packets cannot loop forever. TTL therefore changes at every hop, so option C is always modified.
2. The header checksum is computed over the whole IP header. Because TTL changes at every hop, the checksum value must change too, and the router recomputes it. The checksum is therefore also changed at every hop, so option D is always modified.
3. The Source IP Address stays the same from the sender to the receiver. Only a NAT device rewrites it, and an ordinary forwarding router does not, so option A is not always modified.
4. The Protocol field names the payload carried (for example TCP or UDP). A router does not look at it or change it when forwarding, so option B is not modified.

5. Some fields change only sometimes, such as total length, flags and fragment offset when a router fragments a packet. These do not count as always modified, and none of them is offered as an option.
6. So the fields always modified are TTL and Header Checksum: options C and D.

KEY IDEA

A forwarding router must decrement TTL by 1. The header checksum covers the IP header, so changing TTL forces the router to recompute it. Both therefore change on every hop. The addresses and the Protocol field are left alone.

WATCH OUT FOR

Choosing only TTL and forgetting that the checksum must be recomputed after the TTL change. Another slip is picking Source IP Address because of NAT, which an ordinary router does not do.

Q.33 MSQ · 1 MARK

Programming in C › Pointers & arrays

Consider the following C function definition.

```
int fX(char *a){
  char *b = a;
  while(*b)
     b++;
  return b - a;}
```

Which of the following statements is/are TRUE?

(A) The function call fX("abcd") will always return a value

(B) Assuming a character array c is declared as char c[] = "abcd" in main(), the function call fX(c) will always return a value

(C) The code of the function will not compile

(D) Assuming a character pointer c is declared as char *c = "abcd" in main(), the function call fX(c) will always return a value

Official answer: **A, B, D**

SOLUTION

1. fX(char *a) sets b = a, then repeatedly does b++ while *b is nonzero, i.e., it walks forward until it reaches the string's null terminator '\0'. It returns b - a, the number of characters scanned before that terminator. This is exactly the standard strlen algorithm: it always terminates and always returns a value, provided a points into a properly null-terminated block of memory.
2. Option A: fX("abcd") passes a pointer to the string literal "abcd", which the compiler stores as the 5 bytes 'a','b','c','d','\0'. The loop scans the 4 non-null characters and stops at the terminator, returning 4. It always returns a value. TRUE.
3. Option B: char c[] = "abcd" declared in main() creates a local array initialised with the same 5 bytes 'a','b','c','d','\0'. Passing c to fX decays the array to a char* pointing at its first element, with identical behaviour to option A, always returning 4. TRUE.
4. Option C: the function body is syntactically and semantically valid C: a pointer declaration (char *b = a*), *a while loop dereferencing a char*, pointer subtraction (b - a) whose result is implicitly convertible to the declared return type int. It compiles without error, so 'will not compile' is FALSE.
5. Option D: char *c = "abcd" declared in main() makes c a pointer directly at the string-literal storage 'a','b','c','d','\0', the same guaranteed-null-terminated storage as in option A. fX(c) walks that same 5-byte block and returns 4, always returning a value. TRUE.
6. Options A, B and D are three different ways of handing fX a pointer into a properly null-terminated character block, so all three are always safe and always return the string length (4 for "abcd"); option C is false because the code compiles fine. Final: A, B, D TRUE; C FALSE.

KEY IDEA

`fX` is just `strlen` written out by hand: `b` walks forward from `a` until it hits the terminating `'\0'`, and `b - a` is the number of characters before it. It always returns *some* value as long as `a` points into a properly null-terminated block of memory, which is true for a string literal argument, for a local array initialised with `"..."`, and for a pointer initialised with `"..."`, since in every one of these cases the compiler lays down the characters followed by a `'\0'`.

WATCH OUT FOR

Assuming the loop can run past the end of allocated memory or that `fX` might not terminate. All three constructions (a string literal argument, a local array initialised from a literal, and a pointer initialised from a literal) guarantee a null-terminated block of memory, so the `while(*b)` loop always finds a `'\0'` and the function always returns; there is no case here where it fails to return.

Q.34 NAT · 1 MARK

Discrete Mathematics › Sets & relations

Let P be the partial order defined on the set $\{1, 2, 3, 4\}$ as follows

$$P = \{(x, x) \mid x \in \{1, 2, 3, 4\}\} \cup \{(1, 2), (3, 2), (3, 4)\}$$

The number of total orders on $\{1, 2, 3, 4\}$ that contain P is _____

Official answer: **5**

SOLUTION

1. A total order containing P is a linear extension of P: a listing of $1, 2, 3, 4$ in which, for every pair $(a, b) \in P$ with $a \neq b$, a comes before b. The reflexive pairs (x, x) impose no condition.
2. The non-reflexive pairs give three constraints: 1 before 2, 3 before 2, and 3 before 4. The Hasse diagram is a zigzag $1 < 2 > 3 < 4$.
3. Only 1 and 3 have nothing forced before them, so the first element of the listing is 1 or 3.
4. Case first $= 1$: the remaining elements are $2, 3, 4$. Both 2 and 4 must come after 3, so 3 comes next. The last two places hold $2, 4$ in either order. This gives $1, 3, 2, 4$ and $1, 3, 4, 2$, which is 2 orders.
5. Case first $= 3$: the remaining elements are $1, 2, 4$ and the only constraint left is 1 before 2, with 4 unconstrained. Of the $3! = 6$ arrangements of $\{1, 2, 4\}$, exactly half, 3, have 1 before 2: $3, 1, 2, 4$; $3, 1, 4, 2$; $3, 4, 1, 2$. This is 3 orders.
6. Total $= 2 + 3 = 5$. Check: the two cases are disjoint and every valid listing starts with 1 or 3, so no order is missed or counted twice.

KEY IDEA

Total orders containing a partial order are its linear extensions. Count the permutations that respect every non-reflexive pair, splitting by which element can come first.

WATCH OUT FOR

Counting all $4! = 24$ permutations, or forgetting that a total order must keep P's pairs in the same direction, so 3 must precede both 2 and 4 and 1 must precede 2. Another slip is to treat the reflexive pairs as constraints, or to miss that 4 is free once 3 is placed.

Q.35 NAT · 1 MARK

Data Structures and Algorithms › Design techniques

Let A be an array containing integer values. The distance of A is defined as the minimum number of elements in A that must be replaced with another integer so that the resulting array is sorted in non-decreasing order. The distance of the array $[2, 5, 3, 1, 4, 2, 6]$ is _____.

Official answer: **3**

SOLUTION

1. Reformulate the problem: the elements that are NOT replaced must already appear in non-decreasing order among themselves (as a subsequence), because every replaced position can be filled with any value needed to bridge the gaps and keep the whole array non-decreasing.

2. Therefore the minimum number of replacements equals n minus the length of the Longest Non-Decreasing Subsequence (LNDS) of the array.
3. Let $a = [2, 5, 3, 1, 4, 2, 6]$ (1-indexed), so $n = 7$.
4. Define $L(i)$ = length of the LNDS ending at index i: $L(i) = 1 + \max\{L(j) : j < i,\ a[j] \le a[i]\}$, or $L(i) = 1$ if no such j exists.
5. $L(1) = 1$ for $a_1 = 2$. $L(2) = 2$ for $a_2 = 5$ (using $a_1 = 2$). $L(3) = 2$ for $a_3 = 3$ (using $a_1 = 2$). $L(4) = 1$ for $a_4 = 1$ (no earlier element ≤ 1).
6. $L(5)$ for $a_5 = 4$: eligible predecessors (≤ 4) are $a_1 = 2$ ($L{=}1$), $a_3 = 3$ ($L{=}2$), $a_4 = 1$ ($L{=}1$); taking the maximum gives $L(5) = 1 + 2 = 3$ (subsequence $2, 3, 4$).
7. $L(6)$ for $a_6 = 2$: eligible predecessors (≤ 2) are $a_1 = 2$ ($L{=}1$), $a_4 = 1$ ($L{=}1$); so $L(6) = 1 + 1 = 2$.
8. $L(7)$ for $a_7 = 6$: all earlier elements are ≤ 6, so the eligible set is everything; the maximum among $L(1), \ldots, L(6) = 1, 2, 2, 1, 3, 2$ is 3 (at $i = 5$), giving $L(7) = 1 + 3 = 4$ (subsequence $2, 3, 4, 6$ at indices $1, 3, 5, 7$).
9. The length of the LNDS is $\max_i L(i) = 4$.
10. Distance $= n - \text{LNDS} = 7 - 4 = 3$. Check: keeping positions $1, 3, 5, 7$ (values $2, 3, 4, 6$) and replacing positions $2, 4, 6$ with, say, $2, 3, 3, 4, 4, 4, 6$ gives a valid non-decreasing array using exactly 3 replacements, and no non-decreasing subsequence of a has length greater than 4, so 3 is indeed the minimum.

KEY IDEA
Minimum replacements to make an array non-decreasing = n − (length of the Longest Non-Decreasing Subsequence), because the untouched elements must themselves already form a non-decreasing subsequence, and any gaps left by replaced elements can always be filled with values that bridge them.

WATCH OUT FOR
Comparing the array with its sorted version, [1, 2, 2, 3, 4, 5, 6], and counting the positions that differ gives 5, because only positions 5 and 7 already match. The elements kept unchanged need only be in non-decreasing order among themselves, not in their final sorted positions, so the distance is n minus the length of the longest non-decreasing subsequence: $7 - 4 = 3$.

Q.36 MCQ · 2 MARKS

Programming in C › Pointers & arrays

What is the output of the following C program?

```
 #include <stdio.h>
int main() {
   double a[2]={20.0, 25.0}, *p, *q;
   p = a;
   q = p + 1;
   printf("%d,%d", (int)(q - p), (int)(*q - *p));
   return 0;
}
```

(A) 4,8
(B) 1,5
(C) 8,5
(D) 1,8

Official answer: **B**

SOLUTION
1. double a[2] = {20.0, 25.0} declares an array of 2 doubles: a[0]=20.0, a[1]=25.0.

2. p = a; : array-to-pointer decay makes p point to a[0] (value 20.0).
3. q = p + 1; : pointer arithmetic on a double* advances by one element (the compiler scales by sizeof(double)), so q points to a[1] (value 25.0).
4. q - p is pointer subtraction between two pointers of the same type, which the language defines to yield the number of elements between them, i.e. the byte difference divided by sizeof(double) . Here that is 1 , regardless of sizeof(double) being 8. (int)(q - p) = 1 .
5. *q - *p = 25.0 - 20.0 = 5.0 ; (int)(5.0) = 5 .
6. The printf("%d,%d", ...) therefore prints 1,5 , matching option B.

KEY IDEA

Subtracting two pointers of the same type gives the number of array elements between them (the language defines it as the byte difference divided by the element's `sizeof`), not a raw byte offset. This is different from ordinary pointer-to-integer or byte-address reasoning.

WATCH OUT FOR

Assuming `q - p` returns the byte offset (`sizeof(double) = 8`) rather than the element count (`1`), which leads to wrongly picking option A (4,8) or C (8,5).

Q.37 MCQ · 2 MARKS

Operating Systems › CPU scheduling

Consider a single processor system with four processes A, B, C, and D, represented as given below, where for each process the first value is its arrival time, and the second value is its CPU burst time.

A (0, 10), B (2, 6), C (4, 3), and D (6, 7).

Which one of the following options gives the average waiting times when preemptive Shortest Remaining Time First (SRTF) and Non-Preemptive Shortest Job First (NP-SJF) CPU scheduling algorithms are applied to the processes?

(A) SRTF = 6, NP-SJF = 7

(B) SRTF = 6, NP-SJF = 7.5

(C) SRTF = 7, NP-SJF = 7.5

(D) SRTF = 7, NP-SJF = 8.5

Official answer: **B**

SOLUTION

1. Process table:

Process	Arrival	Burst
A	0	10
B	2	6
C	4	3
D	6	7

   Waiting time $=$ completion $-$ arrival $-$ burst.
2. SRTF timeline: A runs 0 to 2 (remaining 8). At 2, B arrives with 6, less than 8, so B preempts A and runs 2 to 4 (remaining 4). At 4, C arrives with 3, less than B's remaining 4, so C preempts B.
3. C runs 4 to 7. D arrives at 6 with 7, but C has 1 left, so C continues and finishes at 7. Remaining times are A 8, B 4, D 7, so B runs 7 to 11, then D runs 11 to 18, then A runs 18 to 26.
4. SRTF Gantt chart:

A	B	C	B	D	A
0-2	2-4	4-7	7-11	11-18	18-26

5. SRTF waiting times: $W_A = 26 - 0 - 10 = 16$, $W_B = 11 - 2 - 6 = 3$, $W_C = 7 - 4 - 3 = 0$, $W_D = 18 - 6 - 7 = 5$. Total $= 24$, average $= 24/4 = 6$.

6. NP-SJF timeline: A starts at 0 and runs to 10 with no preemption. At 10, B (6), C (3) and D (7) are waiting, so the shortest, C, runs 10 to 13, then B (6) runs 13 to 19, then D runs 19 to 26.
7. NP-SJF waiting times: $W_A = 0$, $W_C = 10 - 4 = 6$, $W_B = 13 - 2 = 11$, $W_D = 19 - 6 = 13$. Total $= 30$, average $= 30/4 = 7.5$.
8. So SRTF $= 6$ and NP-SJF $= 7.5$, which is option (B).

KEY IDEA

In SRTF, re-decide at every arrival: run the process with the least remaining time, and preempt only if the newcomer's burst is strictly less than the running process's remaining time. In NP-SJF, once a process starts it runs to completion, and the next one is the shortest burst among those waiting.

WATCH OUT FOR

Comparing a newly arrived process with the running process's full burst instead of its remaining time. For example, D (7) arrives at 6 while C has only 1 left, so C must not be preempted. Another slip is forgetting to subtract arrival time when computing waiting time.

Q.38 MCQ · 2 MARKS

Computer Networks › IPv4

Which one of the following CIDR prefixes exactly represents the range of IP addresses 10.12.2.0 to 10.12.3.255?

(A) 10.12.2.0/23

(B) 10.12.2.0/24

(C) 10.12.0.0/22

(D) 10.12.2.0/22

Official answer: **A**

SOLUTION

1. Count the addresses in the range 10.12.2.0 to 10.12.3.255. The third octet takes the values 2 and 3, and the fourth octet runs over 0 to 255 for each. That gives $2 \times 256 = 512 = 2^9$ addresses.
2. A CIDR block of 2^9 addresses has 9 host bits, so its prefix length is $32 - 9 = 23$. Only option (A) has a /23 prefix.
3. Check that 10.12.2.0 is a valid /23 network address. In binary the third octet is 00000010. A /23 prefix fixes the first 7 bits of the third octet (0000001), and the last bit is a host bit. That host bit is 0, so all host bits of 10.12.2.0 are zero and it is a valid /23 network address.
4. The block 10.12.2.0/23 covers third octets $0000001x$, that is 2 and 3, with any fourth octet. It therefore spans 10.12.2.0 to 10.12.3.255, which is exactly the given range.
5. Check the other options. (B) 10.12.2.0/24 covers only 256 addresses (10.12.2.0 to 10.12.2.255), so it is too small. (C) 10.12.0.0/22 covers 1024 addresses (10.12.0.0 to 10.12.3.255), so it is too large. (D) 10.12.2.0/22 is also a 1024-address block, so it is too large. It is not even a proper network address, because a /22 block must start at a third octet that is a multiple of 4 (here 10.12.0.0). Only (A) matches the range exactly.

KEY IDEA

The block size 2^k fixes the prefix length $32 - k$. The start address must also have all its host bits zero, which means it must be aligned to a multiple of the block size.

WATCH OUT FOR

Picking a /22 because the range ends at 10.12.3.255 and "looks like" a /22 boundary. A /22 would have to start at 10.12.0.0 and would include 10.12.0.0 to 10.12.1.255, which are outside the range. Another slip is counting only the third octet's two values and answering /24.

Q.39 MCQ · 2 MARKS

Data Structures and Algorithms › Trees & heaps

You are given a set V of distinct integers. A binary search tree T is created by inserting all elements of V one by one, starting with an empty tree. The tree T follows the convention that, at each node, all values stored in the left subtree of the node are smaller than the value stored at the node. You are not aware of the sequence in which these values were inserted into T, and you do not have access to T.

Which one of the following statements is TRUE?

(A) Inorder traversal of T can be determined from V

(B) Root node of T can be determined from V

(C) Preorder traversal of T can be determined from V

(D) Postorder traversal of T can be determined from V

Official answer: **A**

SOLUTION

1. BST invariant: in any binary search tree built from a set V by inserting elements one by one, at every node all values in the left subtree are smaller than the node's value and (implicitly) all values in the right subtree are larger. This holds regardless of insertion order, since it is what defines the BST shape at each step.
2. Inorder traversal visits left subtree, then the node, then right subtree. By induction on subtree size: the inorder traversal of any BST is exactly its elements listed in increasing numeric order. This is a structural property of BSTs, not dependent on how the tree was built.
3. Since V is a fixed set of distinct integers, sorting V (an operation performable knowing only the set V) gives precisely the inorder sequence of T, for whichever insertion order produced T. So inorder traversal is fully determined by V alone: option (A) is TRUE.
4. Root node (B): the root is whichever element of V was inserted first. Different insertion sequences of the same V give different roots, e.g. V = {1,2,3}: inserting order (2,1,3) makes 2 the root, while inserting order (1,2,3) makes 1 the root (with 2 and 3 as right descendants). So the root cannot be pinned down from V alone: (B) is false.
5. Preorder traversal (C): preorder always begins with the root. Since the root varies with insertion order (shown above), the preorder sequence is not fixed by V alone: (C) is false.
6. Postorder traversal (D): postorder always ends with the root, and its full sequence likewise depends on tree shape, which varies with insertion order. The same V = {1,2,3} example gives different postorder sequences for different insertion orders; (D) is false.
7. Only the inorder traversal is invariant across every possible BST shape that V can produce, so the unique TRUE statement is (A).

KEY IDEA

A BST's inorder traversal is always its elements in sorted order. This is a consequence of the left-smaller/right-larger invariant at every node, true for every possible tree shape a set can produce, so it depends only on the set V, never on the insertion sequence that built T. Root, preorder, and postorder all depend on which element was inserted first, which V alone does not reveal.

WATCH OUT FOR

Assuming that because you "don't have access to T" nothing about it can be recovered from V, and picking "none of the traversals is determined" reasoning, or conversely assuming the root (the most naturally guessed structural feature) is fixed by the set; both miss that inorder order is a value-based, shape-independent property while root/preorder/postorder are insertion-order-dependent.

Q.40 MCQ · 2 MARKS

Compiler Design › Parsing

Q.40 Consider the following context-free grammar where the start symbol is S and the set of terminals is $\{a,b,c,d\}$.

$$S \rightarrow AaAb \mid BbBa$$
$$A \rightarrow cS \mid \epsilon$$
$$B \rightarrow dS \mid \epsilon$$

The following is a partially-filled LL(1) parsing table.

	a	b	c	d	\$
S	$S \rightarrow AaAb$	$S \rightarrow BbBa$	(1)	(2)	
A	$A \rightarrow \epsilon$	(3)	$A \rightarrow cS$		
B	(4)	$B \rightarrow \epsilon$		$B \rightarrow dS$	

Which one of the following options represents the CORRECT combination for the numbered cells in the parsing table?

Note: In the options, "blank" denotes that the corresponding cell is empty.

(A) (1) $S \rightarrow AaAb$ (2) $S \rightarrow BbBa$ (3) $A \rightarrow \epsilon$ (4) $B \rightarrow \epsilon$

(B) (1) $S \rightarrow BbBa$ (2) $S \rightarrow AaAb$ (3) $A \rightarrow \epsilon$ (4) $B \rightarrow \epsilon$

(C) (1) $S \rightarrow AaAb$ (2) $S \rightarrow BbBa$ (3) blank (4) blank

(D) (1) $S \rightarrow BbBa$ (2) $S \rightarrow AaAb$ (3) blank (4) blank

Official answer: **A**

SOLUTION

1. Rule for an LL(1) table: put $A \rightarrow \alpha$ in column t if $t \in \text{FIRST}(\alpha)$, or if $\varepsilon \in \text{FIRST}(\alpha)$ and $t \in \text{FOLLOW}(A)$.
2. $\text{FIRST}(AaAb)$: A can start with c or vanish, so we get c and then a. Hence $\text{FIRST}(AaAb) = \{c, a\}$.
3. $\text{FIRST}(BbBa)$: B can start with d or vanish, so we get d and then b. Hence $\text{FIRST}(BbBa) = \{d, b\}$.
4. Cell (1) is $[S, c]$. Since $c \in \text{FIRST}(AaAb)$, it holds $S \rightarrow AaAb$. Cell (2) is $[S, d]$. Since $d \in \text{FIRST}(BbBa)$, it holds $S \rightarrow BbBa$.
5. $\text{FOLLOW}(A)$: in $S \rightarrow AaAb$ the first A is followed by a and the second A by b. So $\{a, b\} \subseteq \text{FOLLOW}(A)$. ($A$ occurs nowhere else on a right-hand side.)
6. Cell (3) is $[A, b]$. Since $b \in \text{FOLLOW}(A)$, we place $A \rightarrow \varepsilon$ there. Column a already shows the same entry.
7. $\text{FOLLOW}(B)$: in $S \rightarrow BbBa$ the first B is followed by b and the second B by a. So $\{a, b\} \subseteq \text{FOLLOW}(B)$.

8. Cell (4) is $[B, a]$. Since $a \in \text{FOLLOW}(B)$, we place $B \to \varepsilon$ there. The given $[B, b]$ entry agrees.
9. So (1) $S \to AaAb$, (2) $S \to BbBa$, (3) $A \to \varepsilon$, (4) $B \to \varepsilon$, which is option A.

KEY IDEA

An LL(1) table entry for $A \to \alpha$ goes under every terminal in $\text{FIRST}(\alpha)$, and, when α can derive ε, also under every terminal in $\text{FOLLOW}(A)$. Here $\text{FIRST}(AaAb) = \{c, a\}$, $\text{FIRST}(BbBa) = \{d, b\}$ and $\text{FOLLOW}(A) = \text{FOLLOW}(B) = \{a, b\}$.

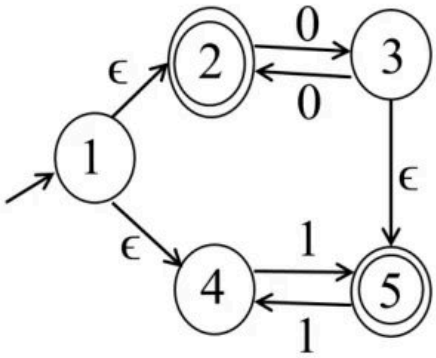
WATCH OUT FOR

Using only FIRST sets and leaving the ε-entries blank (options C and D). The ε-production must also be entered under every terminal of FOLLOW. Another slip is swapping the two S productions, forgetting that a nullable leading symbol makes c or d select the production that begins with A or B respectively.

Q.41 MCQ · 2 MARKS

Theory of Computation › Finite automata

Q.41 Let M be the 5-state NFA with ϵ-transitions shown in the diagram below.



Which one of the following regular expressions represents the language accepted by M ?

(A) $(00)^* + 1(11)^*$

(B) $0^* + (1 + 0(00)^*)(11)^*$

(C) $(00)^* + (1 + (00)^*)(11)^*$

(D) $0^+ + 1(11)^* + 0(11)^*$

Official answer: **B**

SOLUTION

1. From the start state 1, the ε-moves lead to state 2 or state 4, so $L(M)$ is the union of the strings accepted from 2 and from 4.
2. Branch through 2: the states 2 and 3 form a loop on 0 ($2 \to 3 \to 2$). The strings that bring M back to state 2 are $(00)^*$. State 2 is accepting, so $(00)^*$ is accepted.
3. The strings that reach state 3 from 2 are $(00)^*0 = 0(00)^*$. From 3 the ε-move goes to state 5, which is accepting. At 5 the loop $5 \to 4 \to 5$ reads 11, so M can read any $(11)^*$ and still be in 5. This branch adds $0(00)^*(11)^*$.
4. Branch through 4: reading 1 takes 4 to 5, and then $(11)^*$ keeps M in 5. This gives $1(11)^*$. State 4 is not accepting, so a string that ends in 4 is rejected; in particular this branch does not accept ε.
5. Combine: $L = (00)^* + 0(00)^*(11)^* + 1(11)^* = (00)^* + (1 + 0(00)^*)(11)^*$.

6. Rewrite using $0^* = (00)^* + 0(00)^*$. Taking zero repetitions of 11 gives $0(00)^* \subseteq 0(00)^*(11)^*$, which is already in L. So adding $0(00)^*$ to L changes nothing, and $L = 0^* + (1 + 0(00)^*)(11)^*$. This is option B.
7. Check the other options with test strings. A rejects 011, but M accepts it ($1 \to 2 \to 3 \to 5$ on 0, then 11 goes $5 \to 4 \to 5$). C accepts 11 (take $(00)^*$ as ε, then 11), but M does not accept it ($1 \to 4 \to 5 \to 4$ ends in the non-accepting state 4). D rejects ε, but M accepts ε because $1 \to 2$ on an ε-move.

KEY IDEA

Split the NFA at the start state into the branch through state 2 (the 0-loop, then ε into state 5 and the 11-loop) and the branch through state 4 (which needs one 1 to reach state 5). Then merge the two pieces $(00)^*$ and $0(00)^*$ of the first branch into 0^*.

WATCH OUT FOR

Forgetting the ε-move from state 3 to state 5, which gives the strings $0(00)^*(11)^*$, or treating state 4 as accepting. The first mistake leads to option A. The second mistake would wrongly accept strings such as 11, as in option C.

Q.42 MCQ · 2 MARKS

Data Structures and Algorithms › Design techniques

Consider an array `X` that contains `n` positive integers. A subarray of `X` is defined to be a sequence of array locations with consecutive indices.

The C code snippet given below has been written to compute the length of the longest subarray of `X` that contains at most two distinct integers. The code has two missing expressions labelled (P) and (Q).

```
 int first=0, second=0, len1=0, len2=0, maxlen=0;
for (int i=0; i < n; i++) {
 if (X[i] == first) {
  len2++; len1++;
 } else if (X[i] == second) {
  len2++;
  len1 = (P) ;
  second = first;
 } else {
  len2 = (Q) ;
  len1 = 1; second = first;
 }
 if (len2 > maxlen) {
  maxlen = len2;
 }
 first = X[i];
}
```

Which one of the following options gives the CORRECT missing expressions?

(**Hint:** At the end of the `i`-th iteration, the value of `len1` is the length of the longest subarray ending with `X[i]` that contains all equal values, and `len2` is the length of the longest subarray ending with `X[i]` that contains at most two distinct values.)

(A) (P) len1+1 (Q) len2+1

(B) (P) 1 (Q) len1+1

(C) (P) 1 (Q) len2+1

(D) (P) len2+1 (Q) len1+1

Official answer: **B**

SOLUTION

1. Invariant to maintain: len1 = length of the run of equal values ending at X[i]; len2 = length of the longest at-most-two-distinct-value subarray ending at X[i].

2. (P) branch: X[i] == second and (since the first `if` failed) X[i] != first = X[i-1]. So X[i] differs from its immediate predecessor, meaning the equal-value run ending at i has length exactly 1, hence (P) = 1 (not len1+1, which would double-count).
3. (Q) branch: X[i] differs from both first and second, i.e. a brand-new third distinct value appears. The longest at-most-two-distinct subarray ending at i must drop the oldest of the two previous values and keep only the maximal run of the immediately preceding value (length len1) plus the current new element, giving exactly 2 distinct values and length len1+1. So (Q) = len1+1.
4. Full trace with X = [5,5,5,3,5,5] (n=6), using (P)=1, (Q)=len1+1: i=0 (else branch, first/second both start at 0 which never matches a positive integer): len2=1,len1=1,first=5. i=1,2 (X[i]==first=5): len2=2,3; len1=2,3. i=3 (X=3, else branch): len2=len1+1=4, len1=1, second=5, first=3. i=4 (X=5==second): len2=5, len1=(P)=1, second=3, first=5. i=5 (X=5==first): len2=6, len1=2. maxlen reaches 6 = n, matching that the whole array uses only values {5,3}: 2 distinct values throughout.
5. Cross-check with alternating X = [1,2,1,2,1,2] (2 distinct values throughout, so maxlen must reach n=6): with (P)=1, (Q)=len1+1 the len2 sequence is 1,2,3,4,5,6, correctly reaching n.
6. Testing (Q)=len2+1 (option C) on the same alternating trace gives len2 growing 1,2,4,8,... which overshoots n=6, violating the invariant and eliminating C (and by the (P) check, D and A are eliminated too since they don't use (P)=1).
7. Only option B, with (P)=1 and (Q)=len1+1, satisfies both traces and the stated invariant.

KEY IDEA

Track two lengths in parallel (len1 for the run of the single most-recent value, len2 for the window of at most two distinct values) and derive each missing expression from what changes about the "previous" value when the current element repeats the second-to-last distinct value (P) versus introduces a brand-new third value (Q).

WATCH OUT FOR

Assuming len1 always just increments (using len1+1 for (P)) instead of recognizing that hitting the `else if (X[i]==second)` branch means X[i] is different from its immediate predecessor, so the single-value run must restart at 1; symmetrically, mistakenly using len2+1 for (Q) double-counts the old two-distinct window instead of correctly discarding the older of the two previous values and keeping only the immediately preceding run (len1) plus the new element.

Q.43 MCQ · 2 MARKS

Compiler Design › Intermediate code

Q.43 Consider the following expression: $x[i] = (p + r) * -s[i] + u/w$. The following sequence shows the list of triples representing the given expression, with entries missing for triples (1), (3), and (6).

(0)	+	p	r
(1)			
(2)	uminus	(1)	
(3)			
(4)	/	u	w
(5)	+	(3)	(4)
(6)			
(7)	=	(6)	(5)

Which one of the following options fills in the missing entries CORRECTLY?

(A) (1) =[] s i (3) * (0) (2) (6) []= x i

(B) (1) []= s i (3) - (0) (2) (6) =[] x (5)

(C) (1) =[] s i (3) * (0) (2) (6) []= x (5)

(D) (1) []= s i (3) - (0) (2) (6) =[] x i

Official answer: **A**

SOLUTION

1. Apply precedence: unary minus binds tightest, then * and /, then +, then =. The right-hand side is $((p + r) * (-s[i])) + (u/w)$, assigned to $x[i]$.
2. Triple (0) is + p r, which gives $p + r$. This is the left operand of *.
3. Triple (1) must fetch the value of $s[i]$ (an array element read, or rvalue). In triple notation an array read is =[] s i, so (1) is =[] s i. Triple (2), uminus (1), then negates it to give $-s[i]$.
4. Triple (3) must combine the two operands of *, namely $p + r$ (triple (0)) and $-s[i]$ (triple (2)). So (3) is * (0) (2). It is a multiplication, not a subtraction, so options B and D are wrong.
5. Triples (4) and (5) compute u/w and then add it to (3). This matches the printed table, which confirms (3) is the product.
6. Triple (7) is = (6) (5), an assignment of the value (5) to the location named by (6). So (6) must be the store target $x[i]$. In triple notation an array store is []= x i, with operands x and i. Triple (7) then supplies the value, so (6) does not take (5) as an operand. This rules out options B and C, whose (6) uses (5) or =[].

7. Combining the three entries: (1) `=[] s i`, (3) `* (0) (2)`, (6) `[]= x i`. This is option A.

KEY IDEA
In triples, reading an array element is `=[] a i`. Storing into one is `[]= a i`, followed by a separate `= (that triple) value` triple. The operator in (3) comes from the expression: `*` for the product of $(p + r)$ and $-s[i]$.

WATCH OUT FOR
Swapping `=[]` (read) with `[]=` (store), or giving the store triple the value (5) as its operand when the assignment triple (7) already carries it. Also reading the `*` between $(p + r)$ and $-s[i]$ as a minus sign because of the unary minus.

Q.44 MCQ · 2 MARKS

Engineering Mathematics › Probability & statistics

Let x and y be random variables, not necessarily independent, that take real values in the interval $[0, 1]$. Let $z = xy$ and let the mean values of x, y, z be $\bar{x}, \bar{y}, \bar{z}$, respectively. Which one of the following statements is TRUE?

(A) $\bar{z} = \bar{x}\bar{y}$

(B) $\bar{z} \leq \bar{x}\bar{y}$

(C) $\bar{z} \geq \bar{x}\bar{y}$

(D) $\bar{z} \leq \bar{x}$

Official answer: **D**

SOLUTION

1. Since $x \geq 0$ and $0 \leq y \leq 1$, we have $z = xy \leq x$ for every outcome.
2. Expectation preserves inequalities that hold for every outcome, so $E[z] \leq E[x]$, that is $\bar{z} \leq \bar{x}$. This does not use independence, so (D) is always true.
3. Test (A) and (B) with dependent variables. Take $x = y$, equal to 0 or 1 with probability $\frac{1}{2}$ each. Then $\bar{z} = E[x^2] = \frac{1}{2}$ and $\bar{x}\bar{y} = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. So $\bar{z} > \bar{x}\bar{y}$, which makes (A) and (B) false.
4. Test (C). Take x as above and $y = 1 - x$. Then $z = x(1 - x) = 0$, so $\bar{z} = 0$, while $\bar{x}\bar{y} = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. So $\bar{z} < \bar{x}\bar{y}$, which makes (C) false.
5. Only (D) holds for every pair of random variables on $[0, 1]$.

KEY IDEA
Because $0 \leq y \leq 1$ and $x \geq 0$, $xy \leq x$ for every outcome, and expectation keeps that inequality whether or not x and y are independent. Relations between $\bar{z}$ and $\bar{x}\bar{y}$ depend on the covariance, so they can go either way.

WATCH OUT FOR
Assuming $E[xy] = E[x]E[y]$, which holds only for uncorrelated variables. The question says the variables need not be independent, so (A), (B) and (C) each fail for some dependent pair.

Q.45 MCQ · 2 MARKS

Database Management and Warehousing › Relational algebra

The relation schema, Person(pid, city), describes the city of residence for every person uniquely identified by pid (pid is underlined, the key). The following relational algebra operators are available: selection, projection, cross product, and rename.

To find the list of cities where at least 3 persons reside, using the above operators, the minimum number of cross product operations that must be used is

(A) 1

(B) 2

(C) 3

(D) 4

Official answer: **B**

SOLUTION

1. Without grouping or counting operators, "at least 3 persons live in one city" can only be tested by exhibiting three different tuples of Person with the same city. That needs three copies of Person .
2. Rename three copies: $P_1 = \rho_{P_1}(\text{Person})$, $P_2 = \rho_{P_2}(\text{Person})$, $P_3 = \rho_{P_3}(\text{Person})$.
3. Combine them: $P_1 \times P_2 \times P_3$. Joining three relations takes $3 - 1 = 2$ cross products.
4. Select the tuples where the three persons are distinct and share a city:
$$\sigma_{P_1.\text{city}=P_2.\text{city} \wedge P_2.\text{city}=P_3.\text{city} \wedge P_1.\text{pid}\neq P_2.\text{pid} \wedge P_1.\text{pid}\neq P_3.\text{pid} \wedge P_2.\text{pid}\neq P_3.\text{pid}}(P_1 \times P_2 \times P_3)$$
5. Project the city: $\pi_{P_1.\text{city}}(\ldots)$. This returns exactly the cities with at least 3 residents.
6. One cross product combines only two tuples at a time. It can show that 2 persons share a city, but never 3, because every tuple of the result holds only two persons. Selection, projection and rename never combine tuples. So 2 cross products are needed, and 2 are enough. The answer is (B).

KEY IDEA

With only selection, projection, cross product and rename, a count of 3 can be tested only by pairing three renamed copies of the relation. Three copies take two cross products.

WATCH OUT FOR

Answering 3 because there are three persons, or 1 because a single self-join feels enough. A pair of tuples (one cross product) proves only 2 residents. Three renamed copies are joined by 2 cross products, not 3.

Q.46 MSQ · 2 MARKS

Operating Systems › Synchronisation

Q.46 Consider a multi-threaded program with two threads `T1` and `T2`. The threads share two semaphores: `s1` (initialized to 1) and `s2` (initialized to 0). The threads also share a global variable `x` (initialized to 0). The threads execute the code shown below.

```
// code of T1          | // code of T2
wait(s1);              | wait(s1);
x = x+1;               | x = x+1;
print(x);              | print(x);
wait(s2);              | signal(s2);
signal(s1);            | signal(s1);
```

Which of the following outcomes is/are possible when threads `T1` and `T2` execute concurrently?

(A) `T1` runs first and prints 1, `T2` runs next and prints 2

(B) `T2` runs first and prints 1, `T1` runs next and prints 2

(C) `T1` runs first and prints 1, `T2` does not print anything (deadlock)

(D) `T2` runs first and prints 1, `T1` does not print anything (deadlock)

Official answer: **B, C**

SOLUTION

1. Initial values: $s1 = 1$, $s2 = 0$, $x = 0$. Both threads begin with wait(s1), so whichever thread gets s1 first is alone in the section that follows. s1 acts as a lock.
2. Case 1, T1 first. wait(s1) makes $s1 = 0$, then $x = 1$ and T1 prints 1. T1 then calls wait(s2), but $s2 = 0$, so it blocks while still holding s1.
3. T2 now calls wait(s1), and $s1 = 0$, so it blocks too. Only T2 could signal(s2) and only T1 could signal(s1), and each is waiting for the other. This is a deadlock: T1 prints 1 and T2 prints nothing. So (C) is possible.
4. (A) is impossible. If T1 runs first, T2 can never get past wait(s1), because T1 releases s1 only after wait(s2), and only T2 can signal s2. So T2 never prints 2.
5. Case 2, T2 first. wait(s1) makes $s1 = 0$, then $x = 1$ and T2 prints 1. T2 never blocks after this point. signal(s2) makes $s2 = 1$ and signal(s1) makes $s1 = 1$.
6. T1 now runs. wait(s1) gives $s1 = 0$, then $x = 2$ and T1 prints 2. wait(s2) succeeds ($s2$ goes from 1 to 0) and signal(s1) makes $s1 = 1$. Both threads finish, so (B) is possible.
7. (D) is impossible. After T2 runs first, it signals s2 before releasing s1, so by the time T1 can pass wait(s1) the value of s2 is already 1. T1 therefore never blocks on wait(s2) and always prints 2.
8. Possible outcomes: (B) and (C).

KEY IDEA

Trace each possible first thread. Whoever gets `s1` first holds it as a lock. `T1` holds `s1` while it waits on `s2`, and only `T2` can signal `s2`, but `T2` is stuck at `wait(s1)`. That is a deadlock. `T2` signals `s2` before it releases `s1`, so it never blocks and `T1` always finishes after it.

WATCH OUT FOR

Thinking that `T1` first can still let `T2` print 2 (option A), or that `T2` first can leave `T1` stuck on `wait(s2)` (option D). Both miss that `T1` keeps `s1` while it waits, and that `T2` signals `s2` before it releases `s1`.

Q.47 MSQ · 2 MARKS

Engineering Mathematics › Linear algebra

Let A be an $n \times n$ matrix over the set of all real numbers $\mathbb{R}$. Let B be a matrix obtained from A by swapping two rows. Which of the following statements is/are TRUE?

(A) The determinant of B is the negative of the determinant of A

(B) If A is invertible, then B is also invertible

(C) If A is symmetric, then B is also symmetric

(D) If the trace of A is zero, then the trace of B is also zero

Official answer: **A, B**

SOLUTION

1. Swapping two rows is an elementary row operation, so $B = PA$ where P is the identity matrix with those two rows interchanged. Then $\det P = -1$.
2. (A) $\det B = \det P \cdot \det A = -\det A$. TRUE.
3. (B) A is invertible exactly when $\det A \neq 0$. Then $\det B = -\det A \neq 0$, so B is invertible. TRUE.
4. (C) Take the symmetric matrix $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$. Swapping its rows gives $B = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}$. Here $B_{12} = 2 \neq 1 = B_{21}$, so B is not symmetric. FALSE.
5. (D) Take $A = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix}$, whose trace is $1 + (-1) = 0$. Swapping rows gives $B = \begin{bmatrix} 3 & -1 \\ 1 & 2 \end{bmatrix}$, whose trace is $3 + 2 = 5 \neq 0$. FALSE.
6. Only (A) and (B) are true.

KEY IDEA

A row swap multiplies the determinant by -1, so it keeps invertibility. It does not preserve symmetry or the trace, because those depend on which entries sit on the diagonal and across it.

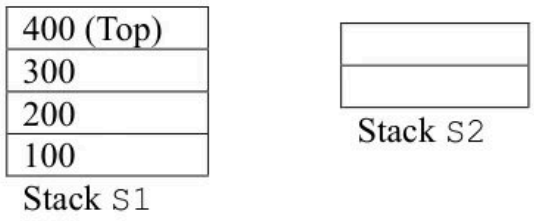
WATCH OUT FOR

Assuming a row swap leaves the matrix's other properties unchanged. Symmetry and trace depend on entry positions, so a single swap can break both.

Q.48 MSQ · 2 MARKS

Data Structures and Algorithms › Linear structures

Q.48 Let S1 and S2 be two stacks. S1 has capacity of 4 elements. S2 has capacity of 2 elements. S1 already has 4 elements: 100, 200, 300, and 400, whereas S2 is empty, as shown below.



Only the following three operations are available:

PushToS2: Pop the top element from S1 and push it on S2.
PushToS1: Pop the top element from S2 and push it on S1.
GenerateOutput: Pop the top element from S1 and output it to the user.

Note that the pop operation is not allowed on an empty stack and the push operation is not allowed on a full stack.

Which of the following output sequences can be generated by using the above operations?

(A) 100, 200, 400, 300

(B) 200, 300, 400, 100

(C) 400, 200, 100, 300

(D) 300, 200, 400, 100

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

SOLUTION

1. Setup: S1 (bottom→top) = 100, 200, 300, 400 (top = 400), S2 is empty with capacity 2. GenerateOutput only removes S1's current top; to output an element buried under k elements, those k elements must first be moved off S1 via PushToS2, and S2 can hold at most 2 at a time.
2. Option (A) 100, 200, 400, 300: outputting 100 first needs 400, 300 and 200 all cleared off S1's top beforehand, with none of them yet output: that means all three would have to sit in S2 at once, but S2's capacity is only 2. So (A) is impossible.
3. Option (B) 200, 300, 400, 100: PushToS2 (400→S2), PushToS2 (300→S2; S2 = [400,300], full) → S1 top is now 200 → GenerateOutput gives 200. PushToS1 (300 back onto S1) → GenerateOutput gives 300. PushToS1 (400 back onto S1) → GenerateOutput gives 400. GenerateOutput gives 100. Result: 200, 300, 400, 100. This matches (B) exactly, so (B) is achievable.
4. Option (C) 400, 200, 100, 300: GenerateOutput gives 400 (S1's initial top). PushToS2 (300→S2) → S1 top is now 200 → GenerateOutput gives 200. GenerateOutput gives 100 (S1 top after 200 removed). PushToS1 (300 back onto S1) → GenerateOutput gives 300. Result: 400, 200, 100, 300. This matches (C) exactly, so (C) is achievable.

5. Option (D) 300, 200, 400, 100: PushToS2 (400→S2) → S1 top is now 300 → GenerateOutput gives 300. GenerateOutput gives 200 (new S1 top). PushToS1 (400 back onto S1) → GenerateOutput gives 400. GenerateOutput gives 100. Result: 300, 200, 400, 100. This matches (D) exactly, so (D) is achievable.
6. Conclusion: (B), (C) and (D) are all achievable output sequences; (A) is not, since it would need 3 elements parked in a capacity-2 auxiliary stack simultaneously.

KEY IDEA

`GenerateOutput` can only remove S1's current top element. To output an element buried under k others, those k elements must first be moved off S1 via `PushToS2`, but S2 holds at most 2 elements at a time, so at most 2 elements can be "parked" and later restored (via `PushToS1`) to reach an element beneath them.

WATCH OUT FOR

Assuming any permutation reachable by "pop some, push some" is achievable without tracking S2's capacity limit of 2. This wrongly admits sequences (like A) that need three elements moved off S1 before the first output.

Q.49 MSQ · 2 MARKS

Digital Electronics › Number systems

Which of the following is/are EQUAL to 224 in radix-5 (i.e., base-5) notation?

(A) 64 in radix-10

(B) 100 in radix-8

(C) 50 in radix-16

(D) 121 in radix-7

Official answer: **A, B, D**

SOLUTION

1. A numeral $d_2d_1d_0$ in radix r has the value $d_2r^2 + d_1r + d_0$.
2. Convert the target: $(224)_5 = 2 \times 25 + 2 \times 5 + 4 = 50 + 10 + 4 = 64$ in decimal.
3. (A) 64 in radix-10 is 64, so it is equal.
4. (B) $(100)_8 = 1 \times 8^2 + 0 \times 8 + 0 = 64$, so it is equal.
5. (C) $(50)_{16} = 5 \times 16 + 0 = 80 \neq 64$, so it is not equal.
6. (D) $(121)_7 = 1 \times 49 + 2 \times 7 + 1 = 49 + 14 + 1 = 64$, so it is equal.
7. The options equal to $(224)_5$ are A, B and D.

KEY IDEA

Convert every number to one common base, here decimal, using positional weights, then compare the values.

WATCH OUT FOR

Reading "50 in radix-16" as 50 in decimal, or forgetting that the digit 5 in base 16 has weight 16 and so gives 80, not 64.

Q.50 MSQ · 2 MARKS

Digital Electronics › Boolean logic

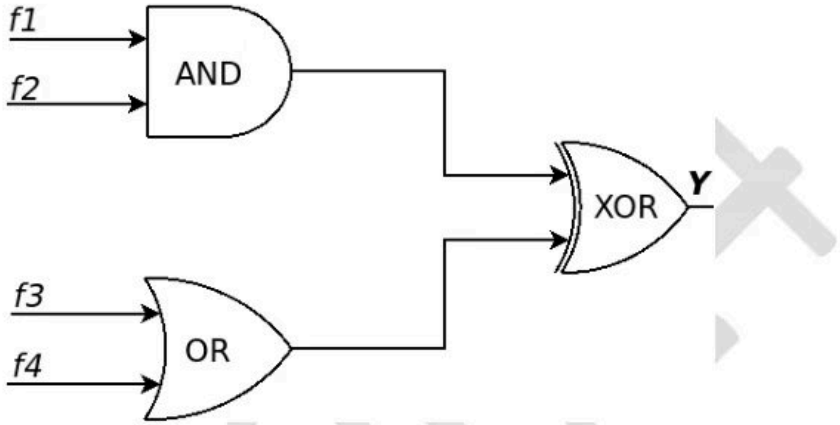
Q.50 Consider 4-variable functions *f1, f2, f3, f4* expressed in sum-of-minterms form as given below.

$$f1 = \sum(0,2,3,5,7,8,11,13)$$

$$f2 = \sum(1,3,5,7,11,13,15)$$

$$f3 = \sum(0,1,4,11)$$

$$f4 = \sum(0,2,6,13)$$



With respect to the circuit given above, which of the following options is/are CORRECT?

(A) $Y = \sum(0,1,2,11,13)$

(B) $Y = \Pi(3,4,5,6,7,8,9,10,12,14,15)$

(C) $Y = \sum(0,1,2,3,4,5,6,7)$

(D) $Y = \Pi(8,9,10,11,12,13,14,15)$

Official answer: **C, D**

SOLUTION

1. The AND gate outputs $f_1 f_2$, which is 1 only on minterms present in both lists. $f_1 = \{0, 2, 3, 5, 7, 8, 11, 13\}$ and $f_2 = \{1, 3, 5, 7, 11, 13, 15\}$, so $f_1 f_2 = \sum m(3, 5, 7, 11, 13)$.
2. The OR gate outputs $f_3 + f_4$, the union of minterms. $f_3 = \{0, 1, 4, 11\}$ and $f_4 = \{0, 2, 6, 13\}$, so $f_3 + f_4 = \sum m(0, 1, 2, 4, 6, 11, 13)$.
3. The XOR gate outputs 1 on a minterm that is in exactly one of the two sets (the symmetric difference). Minterms 11 and 13 are in both sets, so $Y = 0$ there.
4. Only in the AND set: $3, 5, 7$. Only in the OR set: $0, 1, 2, 4, 6$. Hence $Y = \sum m(0, 1, 2, 3, 4, 5, 6, 7)$, so (C) is correct.
5. (A) lists $0, 1, 2, 11, 13$. This keeps 11 and 13, which cancel, and leaves out 3, 5 and 7, so (A) is wrong.
6. $Y = 1$ exactly on minterms 0 to 7, so $Y = 0$ exactly on 8 to 15. As a product of maxterms, $Y = \prod M(8, 9, 10, 11, 12, 13, 14, 15)$, so (D) is correct.

7. (B) lists zeros $3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15$. That would make $Y = 1$ on $\{0, 1, 2, 11, 13\}$, which is the set in (A) and not our Y, so (B) is wrong.

KEY IDEA

XOR of two functions is 1 exactly on the minterms that lie in one of the two sets but not both. Find the AND and OR outputs as minterm sets first, then take the symmetric difference. A function is a sum of its 1-minterms and a product of its 0-maxterms, so the same Y can be written either way.

WATCH OUT FOR

Taking the union or intersection of the AND and OR outputs instead of the symmetric difference, so that 11 and 13 are wrongly kept in Y. This gives option (A). Another slip is writing the product form from the 1-minterms instead of the 0-minterms.

Q.51 MSQ · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Let G be an undirected connected graph in which every edge has a positive integer weight. Suppose that every spanning tree in G has *even* weight. Which of the following statements is/are TRUE for *every* such graph G?

(A) All edges in G have even weight

(B) All edges in G have even weight **OR** all edges in G have odd weight

(C) In each cycle C in G, all edges in C have even weight

(D) In each cycle C in G, either all edges in C have even weight **OR** all edges in C have odd weight

Official answer: **D**

SOLUTION

1. Reduce the problem to edge-weight parities: let $x_e = w(e) \bmod 2$; the hypothesis is that every spanning tree's weight is even, i.e., $\sum_{e \in T} x_e \equiv 0 \pmod 2$ for every spanning tree T.
2. Take any simple cycle C in G and any edge $e_k \in C$. Removing e_k from C leaves a path (a forest), which can always be extended, in a connected graph, to a full spanning tree T of G that does not contain e_k (adding e_k back would close a cycle). Since T is a tree, the unique path between e_k's endpoints inside T must be exactly this path $C - e_k$, so C is precisely the fundamental cycle of e_k with respect to T.
3. For every other edge e_i of C (so $e_i \in T$, lying on that tree path), the standard spanning-tree exchange property gives another spanning tree $T' = T - e_i + e_k$.
4. Compare weights: $\text{weight}(T') = \text{weight}(T) - w(e_i) + w(e_k)$. Since both T and T' have even weight by hypothesis, subtracting mod 2 forces $w(e_i) \equiv w(e_k) \pmod 2$.
5. Since e_k was an arbitrary edge of C, this parity-matching holds between every pair of edges of C: all edges of any cycle C in G share the same parity (all even, or all odd). This is exactly statement (D), proved true for every graph G satisfying the hypothesis.
6. Kill (A) and (C): take a triangle with all three edges of weight 1. Every spanning tree is any 2 of the 3 edges, weight $1 + 1 = 2$ (even); hypothesis holds. But the edges are all odd, not all even (kills A), and the triangle (the only cycle) does not have all-even edges (kills C). This example does not yet refute (B), since all edges here are uniformly odd, which satisfies B's second disjunct.
7. Kill (B): take two triangles sharing exactly one vertex, one with all edges of weight 2 (even), the other with all edges of weight 1 (odd). Any spanning tree must use 2 edges from each triangle (total 4 edges for the 5 vertices), and any such choice gives weight $(2 + 2) + (1 + 1) = 6$, always even; hypothesis holds. But the edges of G are neither uniformly even (the odd triangle breaks this) nor uniformly odd (the even triangle breaks this), so (B) is false here.
8. Since (A), (B), (C) each admit a counterexample while (D) was proved to hold for every valid G, the only statement guaranteed TRUE for every such G is (D).

KEY IDEA

Work with edge weights mod 2. If C is any simple cycle and $e_k \in C$, then $C - e_k$ is a forest, which extends (using connectivity of G) to a spanning tree T not containing e_k; since T is a tree, the unique tree path between the endpoints of e_k is exactly $C - e_k$, so C is precisely the fundamental cycle of e_k with respect to T. For any other edge e_i of C (so $e_i \in T$), the tree $T' = T - e_i + e_k$ is again a spanning tree of G. Since both T and T' must have even weight, $w(e_i) \equiv w(e_k) \pmod 2$. As e_k was an arbitrary edge of C, every edge of C shares the same parity: this is exactly statement (D), and it holds for every graph G satisfying the hypothesis.

WATCH OUT FOR

Assuming the hypothesis forces every edge weight to be even (choice A), or reasoning only about one spanning tree instead of using the exchange property across all spanning trees sharing a fundamental cycle; also mistakenly believing a "same overall parity across all edges" statement (B) follows, when two triangles of different fixed parities (one all-even, one all-odd) sharing a single vertex is a valid counterexample to B while still satisfying the hypothesis.

Q.52 MSQ · 2 MARKS

Theory of Computation › CFG & PDA

Consider a context-free grammar G with the following 3 rules.

$$S \to aS, \quad S \to aSbS, \quad S \to c$$

Let $w \in L(G)$. Let $n_a(w), n_b(w), n_c(w)$ denote the number of times a, b, c occur in w, respectively. Which of the following statements is/are TRUE?

(A) $n_a(w) > n_b(w)$

(B) $n_a(w) > n_c(w) - 2$

(C) $n_c(w) = n_b(w) + 1$

(D) $n_c(w) = n_b(w) * 2$

Official answer: **B, C**

SOLUTION

1. Track how each production changes the counts. $S \to aS$ adds one a and keeps the number of S symbols the same. $S \to aSbS$ adds one a and one b and raises the number of S symbols by 1. $S \to c$ adds one c and lowers the number of S symbols by 1.
2. A derivation starts with one S and ends with none. Let k be the number of uses of $S \to aSbS$, j the number of uses of $S \to aS$, and m the number of uses of $S \to c$. The number of S symbols changes by $+k - m$, so $1 + k - m = 0$, which gives $m = k + 1$.
3. Only $S \to aSbS$ produces a b, so $n_b = k$. Only $S \to c$ produces a c, so $n_c = m = n_b + 1$. Option (C) is TRUE.
4. Both $S \to aS$ and $S \to aSbS$ produce one a each, so $n_a = k + j = n_b + j \geq n_b$.
5. (A): take $w = c$, which is in $L(G)$. Then $n_a = 0$ and $n_b = 0$, so $n_a > n_b$ fails. FALSE.
6. (B): from (C), $n_c - 2 = n_b - 1$. Since $n_a \geq n_b > n_b - 1$, we get $n_a > n_c - 2$ for every $w \in L(G)$. TRUE.
7. (D): for $w = c$, $n_c = 1$ but $n_b * 2 = 0$. FALSE. Since $n_c = n_b + 1$ always holds, $n_c = 2n_b$ could only happen when $n_b = 1$, and it fails for $w = c$ anyway.
8. Hence the TRUE statements are (B) and (C).

KEY IDEA

Count the S symbols during the derivation. Each $S \to aSbS$ adds one S, each $S \to c$ removes one, and the derivation goes from one S to none. So $n_c = n_b + 1$, and since every b is paired with an a, $n_a \geq n_b$.

WATCH OUT FOR

Judging (A) by cases that have at least one b and forgetting that $w = c$ is in the language, where $n_a = n_b = 0$. Another slip is to read $n_a \geq n_b$ as strict, or to treat (D) as true from the single case $n_b = 1$.

Q.53 NAT · 2 MARKS

Computer Organization and Architecture › Memory & cache

Consider a disk with the following specifications: rotation speed of 6000 RPM, average seek time of 5 milliseconds, 500 sectors/track, 512-byte sectors. A file has content stored in 3000 sectors located *randomly* on the disk. Assuming average rotational latency, the total time (in seconds, rounded off to 2 decimal places) to read the entire file from the disk is _____

Official answer: **29.50 to 30.50**

SOLUTION

1. Rotation speed = 6000 RPM = 100 revolutions/second, so time for one full rotation = $\frac{1}{100}\ \text{s} = 10\ \text{ms}$.
2. Average rotational latency = time for half a rotation = $\frac{10}{2} = 5\ \text{ms}$.
3. In one rotation, all 500 sectors of a track pass under the head, so transfer time per sector = $\frac{10\ \text{ms}}{500} = 0.02\ \text{ms}$.
4. The 3000 sectors of the file are located *randomly* on the disk, so consecutive sectors cannot be assumed adjacent: each sector access is an independent disk access requiring its own seek, its own rotational latency, and its own transfer time.
5. Time for one sector access = seek time + rotational latency + transfer time = $5 + 5 + 0.02 = 10.02\ \text{ms}$.
6. Total time for the file = $3000 \times 10.02\ \text{ms} = 30060\ \text{ms} = 30.06\ \text{s}$.

KEY IDEA

When the sectors of a file are scattered randomly across the disk (not contiguous), the disk cannot amortize a single seek and a single rotational delay over the whole transfer. Every sector must be treated as an independent access: its own seek, its own average rotational latency, and its own transfer time, and these per-sector times are simply summed over all sectors in the file.

WATCH OUT FOR

Treating the 3000 sectors as one contiguous read: charging a single seek and a single rotational latency for the whole file and then adding only the total transfer time (3000 × 0.02 ms). That undercounts massively (a few seconds instead of ~30 s) because it ignores that "randomly located" sectors each require their own seek and rotational delay.

Q.54 NAT · 2 MARKS

Computer Networks › TCP

Consider a TCP connection operating at a point of time with the congestion window of size 12 MSS (Maximum Segment Size), when a timeout occurs due to packet loss. Assuming that all the segments transmitted in the next two RTTs (Round Trip Time) are acknowledged correctly, the congestion window size (in MSS) during the third RTT will be _____

Official answer: **4**

SOLUTION

1. A timeout is a severe loss signal. TCP sets the slow-start threshold to half the window at the time of loss: $\text{ssthresh} = 12/2 = 6$ MSS.
2. The congestion window is reset to $\text{cwnd} = 1$ MSS and TCP re-enters slow start.
3. First RTT after the timeout: the window is 1 MSS. The single segment is acknowledged, so the window doubles to 2 MSS for the next RTT.
4. Second RTT: the window is 2 MSS. Both segments are acknowledged, so the window doubles to 4 MSS for the next RTT.
5. Third RTT: the window is 4 MSS. This is below $\text{ssthresh} = 6$, so TCP is still in slow start and there is no linear growth yet.

6. $$\text{cwnd} : 12 \xrightarrow{\text{timeout}} 1 \to 2 \to 4$$
7. So the congestion window during the third RTT is 4 MSS.

KEY IDEA
After a timeout, TCP sets ssthresh to half the old window and restarts slow start from cwnd = 1 MSS, doubling each RTT while below ssthresh: 1, 2, 4 in RTTs 1, 2, 3.

WATCH OUT FOR
Answering 6 (the new ssthresh) or 7 (counting a fast-recovery style halving plus 1), or starting from 12 and growing linearly. A timeout always resets cwnd to 1 MSS, and slow start doubles until ssthresh is reached.

Q.55 NAT · 2 MARKS

Computer Networks › Data link & MAC

Consider an Ethernet segment with a transmission speed of 10^8 bits/sec and a maximum segment length of 500 meters. If the speed of propagation of the signal in the medium is 2×10^8 meters/sec, then the minimum frame size (in bits) required for collision detection is _____

Official answer: **500**

SOLUTION

1. For CSMA/CD the sender must still be transmitting when a collision signal returns from the farthest station. So the transmission time of the frame must be at least the round-trip propagation delay: $T_t \geq 2T_p$.
2. One-way propagation delay: $T_p = \dfrac{500\ \text{m}}{2 \times 10^8\ \text{m/s}} = 2.5 \times 10^{-6}\ \text{s} = 2.5\ \mu\text{s}$.
3. Round-trip time: $2T_p = 5\ \mu\text{s}$.
4. Minimum frame size: $L_{\min} = \text{bandwidth} \times 2T_p = 10^8\ \text{bits/s} \times 5 \times 10^{-6}\ \text{s}$.
5. $$L_{\min} = 500\ \text{bits}$$

KEY IDEA
Collision detection needs the frame's transmission time to be at least the round-trip propagation time, so $L_{\min} = B \times 2T_p$.

WATCH OUT FOR
Using the one-way delay T_p instead of $2T_p$. That gives 250 bits, which is half the correct value.

Q.56 NAT · 2 MARKS

Database Management and Warehousing › Normal forms

A functional dependency $F : X \to Y$ is termed as a *useful functional dependency* if and only if it satisfies all the following three conditions:
- X is not the empty set.
- Y is not the empty set.
- Intersection of X and Y is the empty set.

For a relation R with 4 attributes, the total number of possible useful functional dependencies is _____

Official answer: **50**

SOLUTION

1. A useful dependency $X \to Y$ is determined by an ordered pair of disjoint, non-empty subsets X, Y of the 4 attributes. The condition on $X \cap Y$ means no attribute can be in both.
2. Place each attribute in exactly one of three places: in X, in Y, or in neither. This gives $3^4 = 81$ assignments, and each one gives a disjoint pair (X, Y), with X or Y possibly empty.
3. Assignments with $X = \varnothing$: each attribute is in Y or in neither, so $2^4 = 16$.
4. Assignments with $Y = \varnothing$: each attribute is in X or in neither, so $2^4 = 16$.

5. The single assignment with $X = Y = \varnothing$ (every attribute in neither) is counted in both of the previous steps, so by inclusion-exclusion it must be added back once.
6. $$3^4 - 2^4 - 2^4 + 1 = 81 - 16 - 16 + 1 = 50.$$
7. Check by the size of X: choose $|X| = k$ in $\binom{4}{k}$ ways, then Y is any non-empty subset of the remaining $4 - k$ attributes, in $2^{4-k} - 1$ ways.

$$\sum_{k=1}^{3} \binom{4}{k} \left(2^{4-k} - 1\right) = 4 \cdot 7 + 6 \cdot 3 + 4 \cdot 1 = 28 + 18 + 4 = 50.$$

The term for $k = 4$ is 0 because no attribute is left for Y.

KEY IDEA
Count ordered pairs of disjoint non-empty subsets by giving each attribute one of three places (in X, in Y, in neither), then remove the cases where X or Y is empty by inclusion-exclusion: $3^n - 2 \cdot 2^n + 1$.

WATCH OUT FOR
Forgetting to add back the case where both X and Y are empty after subtracting the two empty cases, which gives 49. Another slip is counting only $2^4 \cdot 2^4$ pairs, which ignores the disjointness condition.

Q.57 NAT · 2 MARKS

Computer Organization and Architecture › Instructions & addressing

A processor with 16 general purpose registers uses a 32-bit instruction format. The instruction format consists of an opcode field, an addressing mode field, two register operand fields, and a 16-bit scalar field. If 8 addressing modes are to be supported, the maximum number of unique opcodes possible for every addressing mode is ____

Official answer: **32**

SOLUTION

1. Total instruction length is fixed at 32 bits, shared among the opcode field, addressing-mode field, two register-operand fields, and a 16-bit scalar (immediate) field.
2. 16 general purpose registers require $\lceil \log_2 16 \rceil = 4$ bits to encode each register operand; with two register-operand fields this uses $2 \times 4 = 8$ bits.
3. The scalar (immediate) field is given directly as 16 bits.
4. Bits used so far (both register fields plus the scalar field) $= 8 + 16 = 24$ bits.
5. Bits remaining for the opcode field and the addressing-mode field together $= 32 - 24 = 8$ bits.
6. 8 addressing modes require $\lceil \log_2 8 \rceil = 3$ bits for the addressing-mode field.
7. Bits left over for the opcode field $= 8 - 3 = 5$ bits.
8. Maximum number of unique opcodes possible for every addressing mode $= 2^5 = 32$.

KEY IDEA
Total instruction width is fixed at 32 bits; every field that is spelled out numerically (register fields, scalar field, addressing-mode field) eats into that budget, and whatever bits remain form the opcode field, whose size determines the count of distinct opcodes as $2^{\text{opcode bits}}$.

WATCH OUT FOR
Forgetting to subtract the addressing-mode field bits before computing the opcode field width, i.e. taking opcode field = 32 − 24 = 8 bits directly (giving 2^8 = 256) instead of first removing the 3 bits needed to encode 8 addressing modes.

Q.58 NAT · 2 MARKS

Computer Organization and Architecture › Pipelining

A non-pipelined instruction execution unit operating at 2 GHz takes an average of 6 cycles to execute an instruction of a program P. The unit is then redesigned to operate on a 5-stage pipeline at 2 GHz. Assume that the ideal throughput of the pipelined unit is 1 instruction per cycle. In the execution of program P, 20% instructions incur an average of 2 cycles stall due to data hazards and 20% instructions incur an average of

3 cycles stall due to control hazards. The speedup (rounded off to one decimal place) obtained by the pipelined design over the non-pipelined design is _____

Official answer: **2.9 to 3.1**

SOLUTION

1. Non-pipelined execution time per instruction: with average CPI = 6 and clock frequency = 2 GHz (cycle time = 0.5 ns), time = 6 × 0.5 ns = 3 ns.
2. Pipelined ideal CPI = 1, since the ideal throughput is given as 1 instruction per cycle.
3. Expected stall cycles per instruction from hazards: 20% of instructions stall an average of 2 cycles (data hazards) → contributes 0.2 × 2 = 0.4 cycles; 20% of instructions stall an average of 3 cycles (control hazards) → contributes 0.2 × 3 = 0.6 cycles. Total expected stall = 0.4 + 0.6 = 1.0 cycle per instruction.
4. Average pipelined CPI = ideal CPI + expected stalls = 1 + 1.0 = 2 cycles per instruction.
5. Pipelined execution time per instruction: same clock frequency (2 GHz, cycle time 0.5 ns), so time = 2 × 0.5 ns = 1 ns.
6. Speedup = (non-pipelined time per instruction) / (pipelined time per instruction) = 3 ns / 1 ns = 3.0. (Equivalently, since both designs share the same clock, speedup = CPI ratio = 6/2 = 3.0.)

KEY IDEA

Speedup from pipelining = (execution time per instruction, non-pipelined) / (execution time per instruction, pipelined). Since both designs run at the same clock frequency, this reduces to the ratio of average CPIs: CPI_non-pipelined / CPI_pipelined, where CPI_pipelined = ideal CPI (1, since throughput is 1 instruction/cycle) plus the expected stall cycles per instruction from all hazards.

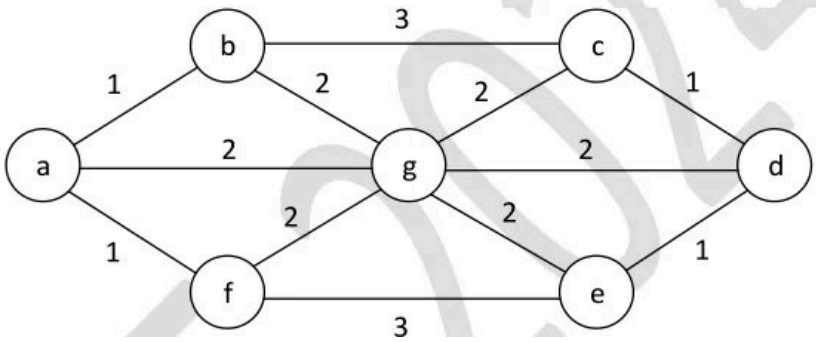
WATCH OUT FOR

Forgetting to add the base ideal CPI of 1 to the stall contributions (using average CPI = 0.4 + 0.6 = 1 instead of 1 + 0.4 + 0.6 = 2), which halves the pipelined execution time and gives a wrong speedup of 6.0; or forgetting that both designs run at the same clock frequency (2 GHz), so speedup can be computed directly as the ratio of average CPIs.

Q.59 NAT · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Q.59 The number of distinct minimum-weight spanning trees of the following graph is _____



Official answer: **9**

SOLUTION

1. List edge weights: weight 1: a-b, a-f, c-d, d-e; weight 2: g-a, g-b, g-c, g-d, g-e, g-f (g joins every other vertex at cost 2); weight 3: b-c, f-e.
2. The four weight-1 edges a-b, a-f, c-d, d-e form no cycle (they build a star at a covering {a,b,f} and a path at d covering {c,d,e}), so they are a valid forest.

3. Each of these four edges is forced into every MST by the standard MST exchange property: for a tree edge candidate of weight 1, if it were excluded from some MST T, the path joining its endpoints inside T would have to use only edges of weight ≤ 1 (otherwise swapping in the weight-1 edge would strictly improve T). Checking each: no all-weight-1 path exists between a and b, between a and f, between c and d, or between d and e using only the other three weight-1 edges (each attempt dead-ends at a vertex with no further weight-1 edge). Hence a-b, a-f, c-d, d-e belong to every minimum spanning tree.
4. After fixing these 4 edges, the graph has collapsed into 3 components: X = {a,b,f}, Y = {c,d,e}, and the isolated vertex g. A spanning tree needs exactly 2 more edges to merge 3 components into 1, and those 2 edges must connect the components without forming a cycle (i.e. they must together touch all of X, Y, g).
5. Only three edges leave g toward X (a-g, b-g, f-g, each weight 2) and three leave g toward Y (c-g, e-g, d-g, each weight 2); the only edges directly between X and Y are b-c and f-e, each weight 3.
6. Enumerate the ways to pick 2 edges that span all three components: (i) one X-g edge + one Y-g edge → cost 2+2 = 4, and this spans X, g, Y; (ii) one X-g edge + one X-Y edge → cost 2+3 = 5; (iii) one Y-g edge + one X-Y edge → cost 2+3 = 5; using two X-Y edges, or two edges both from X to g (or both from Y to g), never touches all three components, so those are not spanning. The minimum extra cost is 4, achieved only by case (i): one edge from g to X and one edge from g to Y.
7. Total minimum weight = (1+1+1+1) from the four forced edges + 2 + 2 = 8.
8. Count the minimum-weight spanning trees: choose 1 of the 3 edges {a-g, b-g, f-g} to attach g to X, and independently 1 of the 3 edges {c-g, e-g, d-g} to attach g to Y. Each of the 3 × 3 = 9 combinations, together with the 4 forced weight-1 edges, gives a distinct acyclic spanning tree of total weight 8, and no other combination reaches weight 8.
9. Hence the number of distinct minimum-weight spanning trees is 3 \times 3 = 9.

KEY IDEA

First pin down every edge that must belong to every MST (via the cut/exchange property; here the four disjoint weight-1 edges), which collapses the graph into a few components; then count only the ways to connect the remaining components at minimum extra cost, treating each such choice as an independent factor.

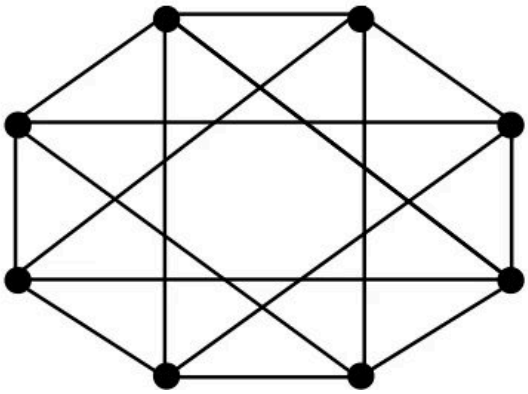
WATCH OUT FOR

Assuming that whichever edges look "cheapest overall" must all be forced, without checking (via the exchange/cut argument) that no equal-or-cheaper alternative path connects their endpoints; or, when counting, forgetting to verify that a candidate pair of connecting edges (e.g. two edges from the same component to g, or two direct X–Y edges) actually spans all remaining components before including it in the count; this can silently over- or under-count the multiplicative total.

Q.60 NAT · 2 MARKS

Discrete Mathematics › Graph theory

Q.60 The chromatic number of a graph is the minimum number of colours used in a *proper* colouring of the graph. The chromatic number of the following graph is _____



Official answer: **2**

SOLUTION

1. Label the vertices $v_0, \ldots, v_7$ around the octagon as in the figure. The edges seen are the outline edges v_iv_{i+1} (indices mod 8), and the chords v_0v_5, v_1v_4, v_2v_7, v_3v_6, v_0v_3, v_1v_6, v_4v_7, v_2v_5. The short top and bottom edges v_0v_1 and v_4v_5 are hard to make out, but the argument below holds whether or not they are present.
2. Check the index difference of every edge. The outline edges differ by 1 (or 7), and the chords differ by 3 (or 5): $0-5$, $1-4$, $2-7$, $3-6$ give $5, 3, 5, 3$, and $0-3$, $1-6$, $4-7$, $2-5$ give $3, 5, 3, 3$. So every edge joins two vertices whose indices differ by an odd number.
3. Hence every edge joins an even-indexed vertex to an odd-indexed vertex. With $A = \{v_0, v_2, v_4, v_6\}$ and $B = \{v_1, v_3, v_5, v_7\}$, no edge lies inside A and none lies inside B. So the graph is bipartite (it is a subgraph of $K_{4,4}$, and equals $K_{4,4}$ when all 16 edges are present).
4. The graph has at least one edge, so one colour is not enough and $\chi(G) \geq 2$.
5. Give every vertex of A colour 1 and every vertex of B colour 2. No edge joins two vertices of the same set, so this is a proper colouring with 2 colours, and $\chi(G) \leq 2$.
6. Therefore $\chi(G) = 2$.

KEY IDEA

A graph is 2-colourable exactly when it has no odd cycle, that is, when it is bipartite. Here the vertices split into even-indexed and odd-indexed ones, and every edge goes between the two groups.

WATCH OUT FOR

Seeing the many crossing lines and assuming a dense graph needs 3 or 4 colours. Density does not matter. The check is whether the vertices split into two groups with no edge inside a group.

Q.61 NAT · 2 MARKS

Computer Organization and Architecture › Instructions & addressing

Q.61 A processor uses a 32-bit instruction format and supports byte-addressable memory access. The ISA of the processor has 150 distinct instructions. The instructions are equally divided into two types, namely R-type and I-type, whose formats are shown below.

R-type Instruction Format:

OPCODE	UNUSED	DST Register	SRC Register1	SRC Register 2

I-type Instruction Format:

OPCODE	DST Register	SRC Register	# Immediate value/address

In the OPCODE, 1 bit is used to distinguish between I-type and R-type instructions and the remaining bits indicate the operation. The processor has 50 architectural registers, and all register fields in the instructions are of equal size.

Let X be the number of bits used to encode the UNUSED field, Y be the number of bits used to encode the OPCODE field, and Z be the number of bits used to encode the immediate value/address field. The value of $X + 2Y + Z$ is

Official answer: **34**

SOLUTION

1. Register field size: with 50 architectural registers, each register field needs $\lceil \log_2 50 \rceil$ bits. Since $2^5 = 32 < 50$ and $2^6 = 64 \geq 50$, each register field (DST, SRC, SRC1, SRC2) is 6 bits.
2. Instruction split: 150 distinct instructions divided equally into R-type and I-type gives 75 instructions of each type.
3. OPCODE size Y: 1 bit of the OPCODE distinguishes R-type from I-type; the remaining $(Y - 1)$ bits must encode the operation within a given type, i.e. address 75 distinct operations. We need $2^{Y-1} \geq 75$. Since $2^6 = 64 < 75$ and $2^7 = 128 \geq 75$, $Y - 1 = 7$, so $Y = 8$.
4. R-type field sum: $Y + X + 6 + 6 + 6 = 32 \Rightarrow 8 + X + 18 = 32 \Rightarrow X = 6$.
5. I-type field sum: $Y + 6 + 6 + Z = 32 \Rightarrow 8 + 12 + Z = 32 \Rightarrow Z = 12$.
6. Compute the answer: $X + 2Y + Z = 6 + 2(8) + 12 = 6 + 16 + 12 = 34$.

KEY IDEA

Each register field must hold $\lceil \log_2 50 \rceil = 6$ bits. The OPCODE's Y bits split into 1 type-bit plus $(Y - 1)$ operation bits, which must distinguish the 75 instructions within EACH type (not all 150 together), so $2^{Y-1} \geq 75 \Rightarrow Y = 8$. The remaining bits in each 32-bit format go to UNUSED (R-type) or the immediate field (I-type).

WATCH OUT FOR

Assuming the OPCODE size only needs to distinguish 150 total instructions (giving Y=8 anyway by coincidence here, but conceptually wrong) or forgetting that the 1 type-bit is already included inside Y, causing double-counting; also using ceil(log2(150))=8 directly on the whole opcode without splitting off the type bit, and separately mis-sizing register fields as ceil(log2(50))=5 or 6 without checking 2^5=32<50.

Q.62 NAT · 2 MARKS

Theory of Computation › Finite automata

Let L_1 be the language represented by the regular expression $b^*ab^*(ab^*ab^*)^*$ and $L_2 = \{w \in (a+b)^* \mid |w| \leq 4\}$, where $|w|$ denotes the length of string w. The number of strings in L_2 which are also in L_1 is _____.

Official answer: **15**

SOLUTION

1. Read the expression $b^*ab^*(ab^*ab^*)^*$. The first part b^*ab^* contains exactly one a. Each repetition of (ab^*ab^*) adds exactly two more a's. The b's can sit anywhere between the a's.
2. So every string in L_1 has an odd number of a's, which means at least one. Conversely, take any string with an odd number of a's. Its first a is matched by the leading ab^* part, and the remaining a's pair up in order. The b's between them are absorbed by the b^* parts. Hence $L_1 = \{w \in \{a,b\}^* \mid w \text{ has an odd number of } a\text{'s}\}$.
3. For $n \geq 1$, exactly half of the 2^n strings of length n have an odd number of a's. To see this, flip the first symbol between a and b. This pairs each even-parity string with an odd-parity one. So the count is 2^{n-1}.
4. For $n = 0$, the empty string ε has zero a's, so it is not in L_1. The count is 0.
5. $n = 1$: 1; $n = 2$: 2; $n = 3$: 4; $n = 4$: 8.
6. Total $= 0 + 1 + 2 + 4 + 8 = 15$.

KEY IDEA

The expression $b^*ab^*(ab^*ab^*)^*$ generates exactly the strings with an odd number of a's. Among strings of each length $n \geq 1$, half have odd parity, so count 2^{n-1} per length.

WATCH OUT FOR

Counting the empty string, or counting only strings with exactly one a or only some of the odd-a strings. Forgetting that the b^* parts let the b's go anywhere is the usual cause.

Q.63 NAT · 2 MARKS

Discrete Mathematics › Groups

Let Z_n be the group of integers $\{0, 1, 2, \ldots, n-1\}$ with addition modulo n as the group operation. The number of elements in the group $Z_2 \times Z_3 \times Z_4$ that are their own inverses is

Official answer: **4**

SOLUTION

1. An element g is its own inverse exactly when $g + g = e$, that is, $2g = e$.
2. The operation in a direct product is componentwise, so (a, b, c) satisfies $2(a, b, c) = (0, 0, 0)$ iff $2a \equiv 0 \pmod 2$, $2b \equiv 0 \pmod 3$ and $2c \equiv 0 \pmod 4$. The three conditions are independent, so the choices can be counted separately.
3. In $\mathbb{Z}_2$: $2a \equiv 0 \pmod 2$ holds for every a, so there are 2 choices $(0, 1)$.
4. In $\mathbb{Z}_3$: $2b \equiv 0 \pmod 3$ forces $b = 0$ because $\gcd(2, 3) = 1$, so there is 1 choice.
5. In $\mathbb{Z}_4$: $2c \equiv 0 \pmod 4$ gives $c \in \{0, 2\}$, so there are 2 choices.
6. Multiply the independent choices: $2 \times 1 \times 2 = 4$.
7. The four elements are $(0,0,0), (1,0,0), (0,0,2), (1,0,2)$.

KEY IDEA
In a direct product of groups the operation acts on each coordinate separately, so the elements with $2g = e$ can be counted coordinate by coordinate. In $\mathbb{Z}_n$ the solutions of $2x \equiv 0 \pmod{n}$ number $\gcd(2, n)$, which gives $2 \cdot 1 \cdot 2 = 4$.

WATCH OUT FOR
Counting only the identity, or taking the self-inverse elements of $\mathbb{Z}_3$ to be $\{0, 1, 2\}$ or the ones of $\mathbb{Z}_4$ to be just $\{0\}$. Another error is adding the counts (2+1+2=5) instead of multiplying them.

Q.64 NAT · 2 MARKS

Operating Systems › Memory management

Consider a 32-bit system with 4 KB page size and page table entries of size 4 bytes each. Assume 1 KB = 2^10 bytes. The OS uses a 2-level page table for memory management, with the page table containing an outer page directory and an inner page table. The OS allocates a page for the outer page directory upon process creation. The OS uses demand paging when allocating memory for the inner page table, i.e., a page of the inner page table is allocated only if it contains at least one valid page table entry.

An active process in this system accesses 2000 unique pages during its execution, and none of the pages are swapped out to disk. After it completes the page accesses, let X denote the minimum and Y denote the maximum number of pages across the two levels of the page table of the process.

The value of X+Y is _____

Official answer: **1028**

SOLUTION

1. Page size is $4\ \text{KB} = 2^{12}$ bytes, so the offset is 12 bits. The virtual page number is $32 - 12 = 20$ bits.
2. One page-table page holds $4096/4 = 1024 = 2^{10}$ entries. The 20-bit page number therefore splits into 10 bits for the outer directory and 10 bits for the inner table.
3. The outer directory is one page, allocated at process creation. It has 1024 entries, so it can point to at most 1024 inner-table pages. Every inner-table page covers 1024 virtual pages.
4. Minimum: one inner-table page can hold valid entries for at most 1024 accessed pages. Since $\lceil 2000/1024 \rceil = 2$, at least 2 inner pages are needed. Two suffice if the 2000 pages are consecutive and lie within two inner tables. So $X = 1 + 2 = 3$.
5. Maximum: an inner page is allocated only if it holds at least one valid entry, and there are 1024 outer entries, so there can be at most 1024 inner pages. This is reachable because $2000 \geq 1024$: place one accessed page under each outer entry, and the remaining $2000 - 1024 = 976$ pages under any of those inner pages. So $Y = 1 + 1024 = 1025$.
6. $X + Y = 3 + 1025 = 1028$.

KEY IDEA
With 4 KB pages and 4-byte entries, each page-table page holds 1024 entries. The minimum comes from packing the accessed pages into as few inner tables as possible, and the maximum from spreading them over as many inner tables as the outer directory can point to. The outer directory is always exactly one page.

WATCH OUT FOR
Taking the maximum as 2000 inner pages. The outer directory has only 1024 entries, so at most 1024 inner pages can exist. Another slip is forgetting to count the one outer-directory page in both X and Y.

Q.65 NAT · 2 MARKS

Compiler Design › Parsing

Consider the following augmented grammar, which is to be parsed with a SLR parser. The set of terminals is {a, b, c, d, #, @}

$S' \rightarrow S$
$S \rightarrow SS \mid Aa \mid bAc \mid Bc \mid bBa$

$A \to d\#$
$B \to @$

Let $I_0 = \text{CLOSURE}(\{S' \to \bullet S\})$. The number of items in the set $\text{GOTO}(I_0, S)$ is _____

Official answer: **9**

SOLUTION

1. $I_0 = \text{CLOSURE}(\{S' \to \bullet S\})$ contains $S' \to \bullet S$, then (dot before S) all S-productions $S \to \bullet SS$, $S \to \bullet Aa$, $S \to \bullet bAc$, $S \to \bullet Bc$, $S \to \bullet bBa$, and (dot before A and B) $A \to \bullet d\#$, $B \to \bullet @$.
2. $\text{GOTO}(I_0, S)$ is the closure of the set obtained by moving the dot over S in every item of I_0 that has the dot immediately before S. Only $S' \to \bullet S$ and $S \to \bullet SS$ qualify.
3. Moving the dot gives the kernel: $S' \to S\bullet$ and $S \to S \bullet S$.
4. In $S \to S \bullet S$ the dot is before the nonterminal S, so add every S-production with the dot at the start: $S \to \bullet SS$, $S \to \bullet Aa$, $S \to \bullet bAc$, $S \to \bullet Bc$, $S \to \bullet bBa$ (5 items).
5. The items $S \to \bullet Aa$ and $S \to \bullet Bc$ have the dot before A and B, so add $A \to \bullet d\#$ and $B \to \bullet @$ (2 items).
6. The items $S \to \bullet bAc$ and $S \to \bullet bBa$ have the dot before the terminal b, so they add nothing. No other nonterminal appears after a dot, so the closure is complete.
7. Count: $2 + 5 + 2 = 9$. The set is $\{S' \to S\bullet,\ S \to S \bullet S,\ S \to \bullet SS,\ S \to \bullet Aa,\ S \to \bullet bAc,\ S \to \bullet Bc,\ S \to \bullet bBa,\ A \to \bullet d\#,\ B \to \bullet @\}$.

KEY IDEA

GOTO(I, X) moves the dot over X in every item that has the dot just before X, then takes the closure. Closure adds productions only for a nonterminal that sits immediately after a dot.

WATCH OUT FOR

Leaving out the kernel item $S' \to S\bullet$, or adding items for A and B from $S \to \bullet bAc$ and $S \to \bullet bBa$. There the dot is before the terminal b, so they add nothing. Another slip is counting the old I_0 items instead of recomputing the closure.

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 2024 Set 2 question paper, organised by IISc Bengaluru; 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 IISc Bengaluru 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