GATE Computer Science and Information Technology

Sample Paper – 8

Duration: 128 Minutes **Maximum Marks: 72**

Instructions

- This paper contains **46 questions** covering the core **Computer Science and Information Technology** subject of the GATE paper conducted by the IITs and IISc (General Aptitude and Engineering Mathematics are provided as separate papers).
- **Q1–Q20 carry 1 mark each** and **Q21–Q46 carry 2 marks each**, for a total of **72 marks**.
- Each question carries **negative marking of one-third of its allotted marks**: a wrong 1-mark question costs **0.33** and a wrong 2-mark question costs **0.67**. Unattempted questions carry no penalty.
- Every question here is a single-correct **MCQ**. The real GATE paper also has Multiple Select (MSQ) and Numerical Answer Type (NAT) questions, and **those carry no negative marking**.
- Attempt this paper in one timed sitting of about **128 minutes**, the share this core subject earns at GATE's overall pace of 180 minutes for 65 questions. Unless stated otherwise, use $\log$ to base 2 for asymptotic work and standard byte $= 8$ bits.

Digital Logic & Computer Organization

Q1. The hexadecimal number $(2\text{AF})_{16}$ is converted to its decimal (base-10) equivalent. The value obtained is: **[1 mark]**

(A) 685

(B) 687

(C) 671

(D) 703

Q2. Using the absorption law of Boolean algebra, the expression $F = A + \overline{A}\,B$ simplifies to: **[1 mark]**

(A) $A \cdot B$

(B) $\overline{A}$

(C) $A + B$

(D) B

Q3. A multiplexer is required to select one output from 16 data inputs. The number of select (control) lines needed by this multiplexer is: **[1 mark]**

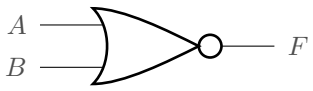
(A) 4

(B) 16

(C) 8

(D) 5

Q4. The single logic gate shown below has two inputs A and B. The Boolean expression for its output F is: **[1 mark]**



NOR gate

(A) $A \cdot B$

(B) $A + B$

(C) $\overline{A + B}$

(D) $\overline{A \cdot B}$

Q5. A 4-bit binary ripple (asynchronous) counter is driven by a clock of frequency 16 kHz. Each flip-flop divides the frequency at its input by two. The frequency at the output of the most-significant flip-flop is: **[1 mark]**

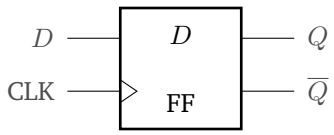
(A) 1 kHz

(B) 2 kHz

(C) 4 kHz

(D) 8 kHz

Q6. The positive-edge-triggered flip-flop shown below has a single data input D. Its next-state characteristic equation Q_{n+1} is: **[1 mark]**



(A) $Q_{n+1} = Q_n$

(B) $Q_{n+1} = \overline{D}$

(C) $Q_{n+1} = D \cdot Q_n$

(D) $Q_{n+1} = D$

Q7. A memory chip is organised as $4\text{K} \times 8$ (that is, 4K addressable locations, each 8 bits wide). The number of address lines required to address every location of this chip is: **[1 mark]**

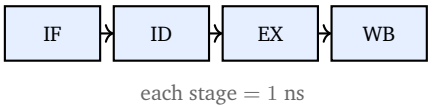
(A) 12

(B) 8

(C) 4096

(D) 13

Q8. A non-pipelined processor takes 4 ns to execute one instruction. It is redesigned as the ideal 4-stage pipeline shown below, in which every stage takes 1 ns. For a very large number of instructions and no stalls, the speedup of the pipelined design over the non-pipelined one approaches: **[1 mark]**



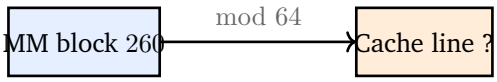
(A) 1

(B) 4

(C) 16

(D) 3

Q9. A direct-mapped cache has 64 lines (blocks). Main memory is divided into blocks of the same size. As illustrated below, main-memory block number 260 is placed into the cache line whose number is: **[1 mark]**



(A) 0

(B) 260

(C) 2

(D) 4

Programming, Data Structures & Algorithms

Q10. Consider the following C statements:

```
int x = 12;    printf("%d", x & (x - 1));
```

The value printed is: **[1 mark]**

(A) 12

(B) 4

(C) 11

(D) 8

Q11. The infix expression

$$a + b * c - d$$

is converted to postfix (reverse-Polish) notation using the usual operator precedence and left-to-right associativity. The resulting postfix expression is: **[1 mark]**

(A) $a\,b\,c\,*\,+\,d\,-$

(B) $a\,b\,+\,c\,*\,d\,-$

(C) $a\,b\,c\,d\,*\,+\,-$

(D) $a\,b\,*\,c\,+\,d\,-$

Q12. A queue (FIFO) is to be implemented using only ordinary stacks (LIFO) as the underlying storage. The minimum number of stacks required to implement one queue is: **[1 mark]**

(A) 1

(B) 2

(C) 3

(D) 4

Q13. A singly linked list of n nodes maintains only a head pointer (there is no tail pointer). The worst-case time complexity of inserting a new node at the end of this list is: **[1 mark]**

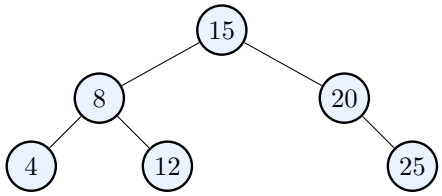
(A) $O(1)$

(B) $O(\log n)$

(C) $O(n \log n)$

(D) $O(n)$

Q14. For the binary search tree shown below, the sequence produced by a *preorder* traversal (root, left subtree, right subtree) is: **[1 mark]**



(A) 15 8 4 12 20 25

(B) 4 8 12 15 20 25

(C) 4 12 8 25 20 15

(D) 15 8 20 4 12 25

Q15. A binary heap with n elements is stored in an array as a complete binary tree. The number of leaf nodes (nodes with no children) in this heap is: **[1 mark]**

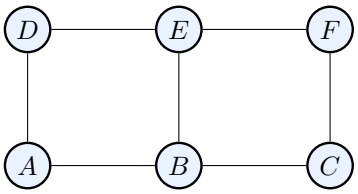
(A) $n/2$

(B) $\lfloor n/2 \rfloor$

(C) $n - 1$

(D) $\lceil n/2 \rceil$

Q16. The connected undirected graph shown below has 6 vertices. The number of edges in *any* spanning tree of this graph is: **[1 mark]**



(A) 6

(B) 15

(C) 7

(D) 5

Q17. Let $f(n) = 3n^2 + 5n + 100$. The tightest asymptotic upper bound (big-O) for $f(n)$ is: **[1 mark]**

(A) $O(n)$

(B) $O(n^2)$

(C) $O(n^3)$

(D) $O(n \log n)$

Q18. The running time of an algorithm satisfies the recurrence

$$T(n) = T\left(\tfrac{n}{2}\right) + 1, \qquad T(1) = 1.$$

The asymptotic solution of this recurrence is: **[1 mark]**

(A) $\Theta(n)$

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

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

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

Q19. Merge sort is applied to an array of n elements. Its worst-case time complexity is: **[1 mark]**

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

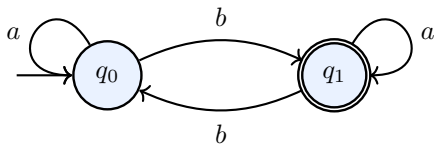
(B) $\Theta(n)$

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

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

Theory of Computation & Compiler Design

Q20. The deterministic finite automaton over $\{a, b\}$ is shown below; q_0 is the start state and q_1 (double circle) is the only accepting state. The language accepted by this DFA is the set of all strings that contain: **[1 mark]**



(A) an even number of b's

(B) the substring bb somewhere

(C) an odd number of b's

(D) strings ending with the symbol a

Q21. Over the alphabet $\{a, b\}$, the regular expression

$$a^* \, b^*$$

describes exactly the set of all strings that: **[2 marks]**

(A) can be any string over $\{a, b\}$

(B) have the form $a^m b^n$ with $m, n \geq 0$ (every a appears before every b)

(C) contain an equal number of a's and b's

(D) end with the substring ab

Q22. Which one of the following languages over $\{a, b\}$ is *regular*? **[2 marks]**

(A) $\{ w \mid$ the number of a's in w is a multiple of $3 \}$

(B) $\{ a^n b^n \mid n \geq 0 \}$

(C) $\{ a^n b^n c^n \mid n \geq 0 \}$

(D) $\{ ww \mid w \in \{a, b\}^* \}$

Q23. The class of context-free languages is closed under some operations but not under others. Context-free languages are *not* closed under which one of the following operations? **[2 marks]**

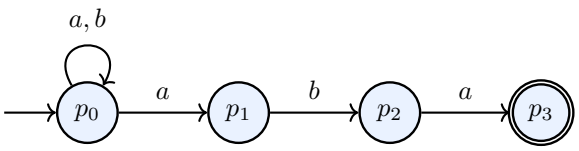
(A) union

(B) concatenation

(C) Kleene star

(D) intersection

Q24. The nondeterministic finite automaton (NFA) shown below has 4 states. When it is converted to an equivalent deterministic finite automaton (DFA) by the subset-construction method, the maximum possible number of states in that DFA is: **[2 marks]**



(A) 4

(B) 8

(C) 5

(D) 16

Q25. The compiler phase that checks whether the source program is meaningful, for example catching the use of an undeclared variable or a type mismatch between operands, is the: [2 marks]

(A) semantic analyzer

(B) lexical analyzer (scanner)

(C) code generator

(D) syntax analyzer (parser)

Q26. In a shift-reduce (bottom-up) LR parser, the data structure used to hold the sequence of states and grammar symbols seen so far while parsing is a: [2 marks]

(A) queue

(B) stack

(C) heap

(D) hash table

Q27. Left factoring is a grammar transformation that eliminates the ambiguity in choosing a production when two alternatives share a common prefix. It is performed mainly to make a grammar suitable for: [2 marks]

(A) LR (bottom-up) parsing

(B) LALR parsing

(C) top-down predictive (LL) parsing

(D) operator-precedence parsing

Q28. A compiler replaces the statement `x = y * 8` with the statement `x = y « 3`, because a left shift is cheaper than a multiplication. This machine-independent optimization is called: [2 marks]

(A) constant folding

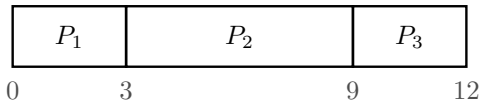
(B) common subexpression elimination

(C) strength reduction

(D) dead-code elimination

Operating Systems & Databases

Q29. Three processes arrive together at time 0 with CPU burst times $P_1 = 3$, $P_2 = 6$ and $P_3 = 3$ (in ms). They are scheduled by non-preemptive First-Come-First-Served (FCFS) in the order P_1, P_2, P_3, whose Gantt chart is shown. The average waiting time of the three processes is: **[2 marks]**



(A) 4 ms

(B) 6 ms

(C) 8 ms

(D) 5 ms

Q30. A counting semaphore S is initialised to 5. A sequence of operations then performs three successful `wait`(S) (P) operations followed by one `signal`(S) (V) operation. The value of S after these four operations is: **[2 marks]**

(A) 5

(B) 2

(C) 3

(D) 4

Q31. A system has 4 processes that share a single resource type, and each process needs a maximum of 2 units of that resource to complete. The minimum number of units of the resource that guarantees the system can never deadlock is: **[2 marks]**

(A) 4

(B) 8

(C) 5

(D) 6

Q32. A system uses 32-bit virtual addresses and a page size of 4 KB. Assuming a single-level page table with one entry per page, the number of entries in the page table is: **[2 marks]**

(A) 2^{12}

(B) 2^{32}

(C) 4096

(D) 2^{20}

Q33. A demand-paging system has 3 page frames, all initially empty, and uses the LRU (Least Recently Used) replacement policy. For the page-reference string shown below, the total number of page faults is: **[2 marks]**

ref:	1	2	3	2	4	1	5

(A) 7

(B) 5

(C) 6

(D) 4

Q34. In a file system that uses a certain block-allocation scheme, each file is stored as blocks scattered anywhere on the disk, and every block holds a pointer to the next block of the same file. This allocation method is called: **[2 marks]**

(A) contiguous allocation

(B) linked allocation

(C) indexed allocation

(D) paged allocation

Q35. In relational algebra, the binary operation that combines tuples from two relations by matching them on their common attributes (keeping only one copy of each shared attribute) is: [2 marks]

(A) natural join ($\bowtie$)

(B) projection (π)

(C) selection (σ)

(D) rename (ρ)

Q36. A relation schema R is in second normal form (2NF) if it is already in first normal form and, in addition: [2 marks]

(A) the left-hand side of every functional dependency is a candidate key

(B) it has no transitive dependency of a non-prime attribute on a key

(C) no non-prime attribute is partially dependent on any candidate key

(D) every attribute holds only atomic (indivisible) values

Q37. Among the ACID properties of a database transaction, the property guaranteeing that a transaction is executed as a single indivisible unit, so that either all of its operations take effect or none of them do, is: [2 marks]

(A) consistency

(B) atomicity

(C) isolation

(D) durability

Q38. In SQL, the keyword that is added to a `SELECT` query to eliminate duplicate rows from its result set is: [2 marks]

(A) `DISTINCT`

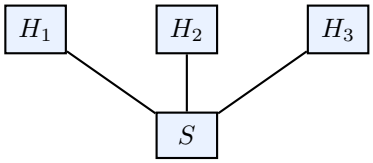
(B) `UNIQUE`

(C) `GROUP BY`

(D) `HAVING`

Computer Networks

Q39. In the local-area network shown below, several hosts are connected through the device labelled S, which forwards frames by looking at their destination MAC (physical) addresses. In the OSI reference model, such a device (a switch/bridge) operates primarily at the: **[2 marks]**



switch S

(A) data-link layer (layer 2)

(B) network layer (layer 3)

(C) physical layer (layer 1)

(D) transport layer (layer 4)

Q40. A cyclic-redundancy-check (CRC) scheme uses the generator polynomial represented by the bit pattern 1011. The number of check bits (the frame check sequence) appended to the data before transmission is: **[2 marks]**

(A) 4

(B) 3

(C) 2

(D) 1

Q41. A Selective-Repeat ARQ protocol uses an n-bit field for sequence numbers, giving 2^n distinct sequence numbers. The maximum allowable sender window size that avoids ambiguity between new and retransmitted frames is: **[2 marks]**

(A) $2^n - 1$

(B) 2^n

(C) 2^{n-1}

(D) $2^{n-1} - 1$

Q42. A communication link has a bandwidth of 1 Mbps and a one-way propagation delay of 10 ms. The bandwidth–delay product of this link (the number of bits that can be "in flight" on it) is: **[2 marks]**

(A) $10{,}000$ bits

(B) $1{,}000$ bits

(C) $100{,}000$ bits

(D) $5{,}000$ bits

Q43. A network with the address block $192.168.5.0/24$ is subnetted using a $/28$ mask. The number of equal-sized subnets produced is: **[2 marks]**

(A) 8

(B) 16

(C) 32

(D) 14

Q44. On a local-area network, a host knows the IP address of a destination on the same subnet but needs the destination's MAC (physical) address to build the frame. The protocol that resolves an IP address to its corresponding MAC address is: **[2 marks]**

(A) DNS

(B) DHCP

(C) ARP

(D) RARP

Q45. Among the standard e-mail protocols, which one lets a client *retrieve and read* messages while keeping them stored on the mail server, so that the same mailbox stays synchronised across several devices? **[2 marks]**

(A) SMTP

(B) IMAP

(C) POP3

(D) FTP

Q46. For ordinary (short) domain-name lookups, the Domain Name System normally uses which transport-layer protocol and well-known port number? **[2 marks]**

(A) TCP, port 53

(B) UDP, port 53

(C) UDP, port 80

(D) TCP, port 25

Detailed Solutions

Q1.

Solution

Concept – hexadecimal to decimal by positional weights: Each hex digit is multiplied by a power of 16 according to its position (rightmost position has weight 16^0).

Step 1 – write the digit values: $2 = 2$, A $= 10$, F $= 15$.

Step 2 – attach positional weights: $(2\text{AF})_{16} = 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0$

Step 3 – evaluate the powers of 16**:** $16^2 = 256$, $16^1 = 16$, $16^0 = 1$.

Step 4 – multiply each term: $2 \times 256 = 512$

$10 \times 16 = 160$

$15 \times 1 = 15$

Step 5 – add the terms: $512 + 160 + 15 = 687$

Final Answer: $(2\text{AF})_{16} = 687 \Rightarrow$ B

Answer: (B) Go Back to Q1

Q2.

Solution

Concept – absorption / redundancy law: The identity $A + \overline{A}\,B = A + B$ removes the redundant $\overline{A}$ from the second term.

Step 1 – expand using distribution: $A + \overline{A}\,B = (A + \overline{A})(A + B)$

Step 2 – simplify the first factor: $A + \overline{A} = 1$

Step 3 – substitute: $= 1 \cdot (A + B)$

Step 4 – conclude: $= A + B$

Why the other options are wrong:

- A, $A \cdot B$: is 1 only when both are 1; but $F = 1$ already when $A = 1$ alone.
- B and D drop a term that is genuinely required.

Final Answer: $F = A + B \Rightarrow$ C

Answer: (C) Go Back to Q2

Q3.

Solution

Concept – select lines of a multiplexer: A multiplexer with 2^k data inputs needs k select lines, because k bits can address 2^k inputs.

Step 1 – set up the equation: $2^k = 16$

Step 2 – express 16 as a power of 2: $16 = 2^4$

Step 3 – match exponents: $2^k = 2^4 \Rightarrow k = 4$

Final Answer: 4 select lines $\Rightarrow$ A

Answer: (A) Go Back to Q3

Q4.

Solution

Concept – reading a NOR gate: A NOR gate is an OR gate followed by an inverting bubble, so it produces the complement of the OR of its inputs.

Step 1 – write the OR of the inputs: $A + B$

Step 2 – apply the output inversion (the bubble): $F = \overline{A + B}$

Step 3 – verify with one row: For $A = 0, B = 0$: $A + B = 0$, so $F = \overline{0} = 1$, which is the defining property of NOR (output high only when all inputs are low).

Why the other options are wrong:

- A, $A \cdot B$: is an AND gate.
- B, $A + B$: is a plain OR, with no inversion.
- D, $\overline{A \cdot B}$: is a NAND gate.

Final Answer: $F = \overline{A + B} \Rightarrow$ C

Answer: (C) Go Back to Q4

Q5.

Solution

Concept – frequency division in a ripple counter: Each flip-flop toggles once per two input pulses, so it halves the frequency. With m cascaded flip-flops the output frequency is the clock divided by 2^m.

Step 1 – list the data: input clock $= 16$ kHz, number of flip-flops $m = 4$.

Step 2 – compute the total division factor: $2^m = 2^4 = 16$

Step 3 – divide the clock frequency: $\frac{16\text{ kHz}}{16} = 1\text{ kHz}$

Final Answer: 1 kHz at the MSB output $\Rightarrow$ A

Answer: (A) Go Back to Q5

Q6.

Solution

Concept – D flip-flop characteristic equation: A D flip-flop simply transfers the value at its D input to the output at the active clock edge.

Step 1 – recall the D truth table: if $D = 0$ then $Q_{n+1} = 0$; if $D = 1$ then $Q_{n+1} = 1$.

Step 2 – read off the pattern: in both rows the next state equals the input D and does not depend on the present state Q_n.

Step 3 – write the characteristic equation: $Q_{n+1} = D$

Final Answer: $Q_{n+1} = D \Rightarrow$ D

Answer: (D) Go Back to Q6

Q7.

Solution

Concept – address lines for a memory of given depth: To address N distinct locations you need $\lceil \log_2 N \rceil$ address lines. The data width (8 bits here) does not affect the address-line count.

Step 1 – find the number of locations: $4\text{K} = 4 \times 1024 = 4096$ locations.

Step 2 – express 4096 **as a power of** 2**:** $4096 = 2^{12}$

Step 3 – take the base-2 **logarithm:** $\log_2(2^{12}) = 12$ address lines.

Why the other options are wrong:

- B, 8: is the data-bus width, not the address-line count.
- C, 4096: is the number of locations, not the number of lines.

Final Answer: 12 address lines $\Rightarrow$ A

Answer: (A) Go Back to Q7

Q8.

Solution

Concept – ideal pipeline speedup: Speedup $= \dfrac{\text{time without pipeline}}{\text{time with pipeline}}$. For a large number of instructions the ideal speedup approaches the number of stages k.

Step 1 – non-pipelined time per instruction: $T_{\text{non}} = 4$ ns.

Step 2 – pipelined throughput (steady state): one instruction completes every clock cycle, and one cycle $= 1$ ns, so $T_{\text{pipe}} = 1$ ns per instruction.

Step 3 – compute the speedup: Speedup $= \dfrac{4 \text{ ns}}{1 \text{ ns}} = 4$

Step 4 – confirm with the general rule: ideal speedup $\rightarrow k = 4$ stages, which agrees.

Final Answer: speedup $= 4 \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q8

Q9.

Solution

Concept – line number in a direct-mapped cache: A memory block B maps to the cache line $B \bmod$ (number of cache lines).

Step 1 – list the data: block number $B = 260$, number of cache lines $= 64$.

Step 2 – divide to find the multiple of 64 below 260: $64 \times 4 = 256$

Step 3 – take the remainder: $260 - 256 = 4$

Step 4 – state the mapping: $260 \bmod 64 = 4$, so the block lands in cache line 4.

Final Answer: cache line $4 \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q9

Q10.

Solution

Concept – the bit trick $x \,\&\, (x-1)$: Subtracting 1 flips the lowest set bit of x to 0 and turns all bits below it to 1; the AND then clears that lowest set bit.

Step 1 – write x in binary: $12 = 1100_2$

Step 2 – compute $x - 1$ in binary: $12 - 1 = 11 = 1011_2$

Step 3 – perform the bitwise AND:
$$\begin{array}{r} 1100 \\ \& \; 1011 \\ \hline 1000 \end{array}$$

Step 4 – convert the result to decimal: $1000_2 = 8$

Final Answer: the program prints $8 \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q10

Q11.

Solution

Concept – infix to postfix, honouring precedence: $*$ binds tighter than $+$ and $-$; operators of equal precedence associate left to right. In postfix the operator follows its two operands.

Step 1 – highest-precedence operation first ($b * c$): $b * c \rightarrow b\,c*$

Step 2 – apply the leftmost $+$ ($a + (b * c)$): $a + (b\,c*) \rightarrow a\,b\,c\,*\,+$

Step 3 – apply the final $-$ (subtract d): $(a\,b\,c\,*\,+) - d \rightarrow a\,b\,c\,*\,+\,d\,-$

Step 4 – read the completed postfix string: $a\,b\,c\,*\,+\,d\,-$

Final Answer: $a\,b\,c\,*\,+\,d\,- \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q11

Q12.

Solution

Concept – simulating a queue with stacks: A single stack reverses order, which is the wrong behaviour for FIFO. Two stacks together restore FIFO order.

Step 1 – use an "in" stack for enqueue: every new element is pushed onto stack S_1.

Step 2 – use an "out" stack for dequeue: when a dequeue is requested and S_2 is empty, pop all of S_1 into S_2, which reverses the order so the oldest element is now on top of S_2.

Step 3 – serve dequeues from S_2: popping S_2 returns elements in first-in-first-out order.

Step 4 – conclude the minimum: one stack cannot achieve FIFO, but this two-stack construction does, so the minimum is 2.

Final Answer: 2 stacks $\Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q12

Q13.

Solution

Concept – reaching the tail of a singly linked list: With only a head pointer and no tail pointer, appending at the end requires walking to the last node first.

Step 1 – traverse to the last node: start at the head and follow `next` pointers until a node whose `next` is `NULL` is found; this visits up to n nodes.

Step 2 – link the new node: set the last node's `next` to the new node (a constant-time step).

Step 3 – combine the costs: the traversal dominates and takes $O(n)$ time in the worst case; the single link update is $O(1)$.

Step 4 – state the worst-case bound: overall time $= O(n)$.

Final Answer: $O(n) \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q13

Q14.

Solution

Concept – preorder traversal: Preorder visits the root first, then recursively the left subtree, then the right subtree.

Step 1 – visit the root: 15.

Step 2 – traverse the left subtree of 15 (rooted at 8): visit 8, then its left child 4, then its right child 12.

Left part $= 8,\ 4,\ 12$.

Step 3 – traverse the right subtree of 15 (rooted at 20): visit 20, then its right child 25 (20 has no left child).

Right part $= 20,\ 25$.

Step 4 – concatenate root, left, right: $15,\ 8,\ 4,\ 12,\ 20,\ 25$.

Final Answer: $15\ 8\ 4\ 12\ 20\ 25 \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q14

Q15.

Solution

Concept – leaf count in a complete binary tree (heap): In a heap stored in an array of n elements, indices $\lfloor n/2 \rfloor + 1, \ldots, n$ have no children, so they are the leaves. Their count is $\lceil n/2 \rceil$.

Step 1 – locate the internal nodes: indices 1 through $\lfloor n/2 \rfloor$ each have at least one child, so there are $\lfloor n/2 \rfloor$ internal nodes.

Step 2 – subtract from the total: number of leaves $= n - \lfloor n/2 \rfloor$.

Step 3 – simplify the expression: $n - \lfloor n/2 \rfloor = \lceil n/2 \rceil$.

Step 4 – spot check with $n = 7$: $\lceil 7/2 \rceil = 4$ leaves, and indeed a 7-node heap has leaves at indices $4, 5, 6, 7$. Correct.

Final Answer: $\lceil n/2 \rceil$ leaves $\Rightarrow$ D

Answer: (D) Go Back to Q15

Q16.

Solution

Concept – edges in a spanning tree: A spanning tree of a connected graph with V vertices contains exactly $V - 1$ edges, regardless of how many edges the original graph has.

Step 1 – count the vertices: the graph has $V = 6$ vertices (A, B, C, D, E, F).

Step 2 – apply the tree-edge formula: edges in spanning tree $= V - 1 = 6 - 1$

Step 3 – evaluate: $= 5$

Why the other options are wrong:

- A, 6: equals V, but a tree on V vertices has $V - 1$ edges (a 6-edge subgraph would contain a cycle).
- B, 15: is the edge count of the complete graph K_6, not a tree.
- C, 7: is the number of edges in this particular graph, not in a spanning tree.

Final Answer: 5 edges $\Rightarrow$ D

Answer: (D) Go Back to Q16

Q17.

Solution

Concept – big-O keeps the dominant term: For a polynomial, the highest-degree term determines the asymptotic growth; lower-degree terms and constant coefficients are ignored.

Step 1 – identify the terms of $f(n)$: $3n^2$ (degree 2), $5n$ (degree 1), 100 (degree 0).

Step 2 – pick the highest-degree term: the n^2 term dominates as $n \to \infty$.

Step 3 – drop the constant factor: $3n^2$ is $O(n^2)$.

Step 4 – state the tight bound: $f(n) = O(n^2)$.

Final Answer: $O(n^2) \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q17

Q18.

Solution

Concept – Master theorem on $T(n) = aT(n/b) + f(n)$: Here $a = 1$, $b = 2$, and $f(n) = 1$ (a constant), which is the recurrence of binary search.

Step 1 – compute the critical exponent: $\log_b a = \log_2 1 = 0$, so $n^{\log_b a} = n^0 = 1$.

Step 2 – compare $f(n)$ with $n^{\log_b a}$: $f(n) = 1 = \Theta(n^0)$, so this is Master-theorem Case 2 (they match).

Step 3 – apply Case 2: $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$.

Step 4 – sanity check by unrolling: $T(n)$ makes one call on half the input each level, giving $\log_2 n$ levels of $O(1)$ work, i.e. $\Theta(\log n)$.

Final Answer: $\Theta(\log n) \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q18

Q19.

Solution

Concept – merge sort always splits in half: Merge sort divides the array into two equal halves and merges them in linear time, so its recurrence is $T(n) = 2T(n/2) + \Theta(n)$ in every case.

Step 1 – write the recurrence: $T(n) = 2\,T(n/2) + \Theta(n)$.

Step 2 – apply the Master theorem: $a = 2$, $b = 2$, so $n^{\log_b a} = n^1 = n$; since

$f(n) = \Theta(n)$ matches, this is Case 2.

Step 3 – solve: $T(n) = \Theta(n \log n)$.

Step 4 – note the "worst case": the split is balanced no matter what the input is, so the worst case is also $\Theta(n \log n)$ (unlike quicksort, which can degrade to $\Theta(n^2)$).

Final Answer: $\Theta(n \log n) \Rightarrow$ D

Answer: (D) Go Back to Q19

Q20.

Solution

Concept – track the parity of b's across the two states: State q_0 (start) means an even number of b's so far; state q_1 (accepting) means an odd number of b's so far.

Step 1 – effect of reading a: each a is a self-loop on the current state, so it does not change the count of b's.

Step 2 – effect of reading b: each b switches between q_0 and q_1, i.e. it flips the parity of the number of b's.

Step 3 – identify the accepting condition: the machine ends in q_1 exactly when an *odd* number of b's has been read.

Step 4 – state the language: $L = \{\, w \in \{a, b\}^* \mid w$ has an odd number of b's $\}$.

Final Answer: strings with an odd number of b's $\Rightarrow$ C

Answer: (C) Go Back to Q20

Q21.

Solution

Concept – meaning of a^*b^*: a^* generates zero or more a's and b^* generates zero or more b's; their concatenation forces all a's to come before all b's.

Step 1 – expand a^*: matches $\varepsilon, a, aa, aaa, \ldots$ (any run of a's, possibly none).

Step 2 – expand b^*: matches $\varepsilon, b, bb, bbb, \ldots$ (any run of b's, possibly none).

Step 3 – concatenate the two runs: every string is some a's followed by some b's: $a^m b^n$ with $m, n \geq 0$.

Step 4 – rule out "any string": a string like ba is *not* matched, because a b appears before an a; hence the language is not all of $\{a, b\}^*$.

Final Answer: strings of the form $a^m b^n$, $m, n \geq 0 \Rightarrow$ B

Answer: (B) Go Back to Q21

Q22.

Solution

Concept – when a counting condition stays regular: Counting a symbol *modulo* a fixed constant needs only finitely many states, so it is regular; matching two unbounded counts (like $a^n b^n$) is not.

Step 1 – analyse option A: "number of a's is a multiple of 3" can be recognised by a DFA that cycles through 3 residue states ($0, 1, 2$ modulo 3), so it is regular.

Step 2 – reject option B: $\{a^n b^n\}$ needs to remember an unbounded n; it is context-free but not regular (pumping lemma).

Step 3 – reject option C: $\{a^n b^n c^n\}$ is not even context-free (it is context-sensitive).

Step 4 – reject option D: $\{ww\}$ requires matching two arbitrary halves and is not context-free.

Final Answer: the multiple-of-3 language is regular $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q22

Q23.

Solution

Concept – closure properties of context-free languages: Context-free languages are closed under union, concatenation and Kleene star, but they are *not* closed under intersection (or complement).

Step 1 – confirm closure under the first three: if L_1, L_2 are context-free, grammars for $L_1 \cup L_2$, $L_1 L_2$ and L_1^* can be built directly from their grammars.

Step 2 – give a counterexample for intersection: $L_1 = \{a^n b^n c^m\}$ and $L_2 = \{a^m b^n c^n\}$ are both context-free, but $L_1 \cap L_2 = \{a^n b^n c^n\}$, which is not context-free.

Step 3 – conclude: intersection can take two context-free languages outside the class, so CFLs are not closed under intersection.

Final Answer: not closed under intersection $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q23

Q24.

Solution

Concept – subset (powerset) construction: Each DFA state corresponds to a *subset* of the NFA's states, so an NFA with k states can produce at most 2^k DFA states.

Step 1 – count the NFA states: the NFA has $k = 4$ states (p_0, p_1, p_2, p_3).

Step 2 – count the possible subsets: number of subsets of a 4-element set $= 2^4$.

Step 3 – evaluate: $2^4 = 16$.

Step 4 – interpret "maximum": the equivalent DFA can have at most 16 states (fewer after removing unreachable states, but 16 is the upper bound).

Final Answer: at most 16 DFA states $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q24

Q25.

Solution

Concept – role of the semantic analyzer: After tokens (lexical) and the parse tree (syntax) are built, the semantic analyzer checks the *meaning*: type compatibility, scope rules and declaration of identifiers.

Step 1 – match the described checks: "undeclared variable" and "type mismatch" are meaning-level (semantic) errors, not spelling or grammar errors.

Step 2 – eliminate the others:

- Lexical analyzer: only forms tokens from characters.
- Parser: only checks that the token sequence follows the grammar.
- Code generator: emits target code after all checks pass.

Step 3 – conclude: these checks belong to the semantic analyzer.

Final Answer: semantic analyzer $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q25

Q26.

Solution

Concept – the parser's working store: Both LR (shift-reduce) and predictive parsers are stack-based. In an LR parser, states and grammar symbols are pushed and popped on a stack.

Step 1 – recall the LR mechanism: a "shift" pushes a state, and a "reduce" pops the symbols of the right-hand side and pushes the state for the left-hand nonterminal.

Step 2 – identify the required discipline: these push/pop operations follow last-in-first-out order, which is exactly a stack.

Step 3 – eliminate the others: a queue is FIFO (wrong order); a heap is for priorities; a hash table is for lookups, none of which model the parser's nesting.

Final Answer: a stack $\Rightarrow$ B

Answer: (B) Go Back to Q26

Q27.

Solution

Concept – purpose of left factoring: When two productions of a nonterminal share a common prefix, a top-down predictive parser cannot decide which to use by looking at one token. Left factoring defers the choice until after the common prefix.

Step 1 – recall the transformation: $A \to \alpha\beta_1 \mid \alpha\beta_2$ becomes $A \to \alpha A'$ and $A' \to \beta_1 \mid \beta_2$.

Step 2 – see what it enables: now the parser reads the common prefix α first, then chooses between β_1 and β_2, which makes single-token lookahead (LL) prediction possible.

Step 3 – match the parsing style: this is required for top-down predictive (LL) parsing; bottom-up LR parsers do not need it.

Final Answer: top-down predictive (LL) parsing $\Rightarrow$ C

Answer: (C) Go Back to Q27

Q28.

Solution

Concept – strength reduction: Strength reduction replaces an expensive operation with an equivalent cheaper one, such as turning a multiplication into a shift or an addition.

Step 1 – read the change: `y * 8` (a multiply) is rewritten as `y << 3` (a left shift by 3).

Step 2 – verify equivalence: shifting left by 3 multiplies by $2^3 = 8$, so the two expressions compute the same value.

Step 3 – classify the optimization: swapping a costly multiply for a cheap shift is the textbook definition of strength reduction.

Why the other options are wrong:

- Constant folding evaluates constant expressions at compile time.
- Common subexpression elimination reuses an already-computed value.
- Dead-code elimination removes results that are never used.

Final Answer: strength reduction $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q28

Q29.

Solution

Concept – waiting time under FCFS: Waiting time = (time the process starts running) − (its arrival time). With all arrivals at 0, each process waits for the sum of the bursts ahead of it.

Step 1 – read the start times from the Gantt chart: P_1 starts at 0, P_2 starts at 3, P_3 starts at 9.

Step 2 – compute each waiting time: $W(P_1) = 0 - 0 = 0$

$W(P_2) = 3 - 0 = 3$

$W(P_3) = 9 - 0 = 9$

Step 3 – sum the waiting times: $0 + 3 + 9 = 12$

Step 4 – divide by the number of processes: $\frac{12}{3} = 4$ ms

Final Answer: average waiting time $= 4$ ms $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q29

Q30.

Solution

Concept – semaphore counting: Each successful `wait` (P) decrements the semaphore by 1; each `signal` (V) increments it by 1.

Step 1 – start value: $S = 5$.

Step 2 – apply three `wait` operations: $5 - 1 - 1 - 1 = 2$

Step 3 – apply one `signal` operation: $2 + 1 = 3$

Step 4 – state the final value: $S = 3$.

Final Answer: $S = 3 \Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q30

Q31.

Solution

Concept – deadlock-free resource bound: With P processes each needing a maximum of M units of one resource type, deadlock is impossible if the total number of units R satisfies $R \geq P(M-1)+1$. This guarantees at least one process can always finish.

Step 1 – list the data: $P = 4$ processes, $M = 2$ units maximum each.

Step 2 – give every process one short of its need: worst case each holds $M-1 = 1$ unit: $4 \times 1 = 4$ units are tied up with none able to proceed.

Step 3 – add one more unit to break the standoff: $4 + 1 = 5$ units guarantee that some process can obtain its last unit, finish, and release.

Step 4 – confirm with the formula: $R = P(M-1) + 1 = 4(2-1) + 1 = 5$.

Final Answer: 5 units $\Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q31

Q32.

Solution

Concept – number of page-table entries: A single-level page table has one entry per virtual page, so the count equals (size of virtual address space) $\div$ (page size).

Step 1 – size of the virtual address space: 32-bit addresses give 2^{32} bytes.

Step 2 – size of one page: 4 KB $= 2^{12}$ bytes.

Step 3 – divide to get the number of pages: $\frac{2^{32}}{2^{12}} = 2^{32-12} = 2^{20}$

Step 4 – interpret: there are 2^{20} pages, hence 2^{20} page-table entries.

Final Answer: 2^{20} entries $\Rightarrow$ D

Answer: (D) Go Back to Q32

Q33.

Solution

Concept – LRU replacement: On a miss with all frames full, LRU evicts the page that was used least recently. Track the frame contents and the recency order after each reference.

Step 1 – references $1, 2, 3$ (frames start empty): $1 \rightarrow$ fault $\{1\}$; $2 \rightarrow$ fault $\{1, 2\}$; $3 \rightarrow$ fault $\{1, 2, 3\}$. (3 faults)

Step 2 – reference 2: 2 is already present $\Rightarrow$ hit; recency order becomes 1 (oldest), 3, 2 (newest).

Step 3 – reference 4: miss, evict the LRU page $1 \Rightarrow$ frames $\{3, 2, 4\}$; order: $3, 2, 4$. (fault)

Step 4 – reference 1: miss, evict the LRU page $3 \Rightarrow$ frames $\{2, 4, 1\}$; order: $2, 4, 1$. (fault)

Step 5 – reference 5: miss, evict the LRU page $2 \Rightarrow$ frames $\{4, 1, 5\}$. (fault)

Step 6 – total the faults: 3 (Step 1) $+ 1 + 1 + 1 = 6$ faults (with a single hit at reference 2).

Final Answer: 6 page faults $\Rightarrow$ C

Answer: (C) Go Back to Q33

Q34.

Solution

Concept – linked file allocation: In linked allocation, a file's blocks are chained: each block stores the data plus a pointer to the next block, so the blocks may lie anywhere on the disk.

Step 1 – match the description: "blocks scattered anywhere, each pointing to the next" is precisely linked allocation.

Step 2 – eliminate the others:

- Contiguous: all blocks occupy consecutive disk locations.

- Indexed: one index block holds all the pointers; the data blocks themselves do not chain.
- "Paged allocation" is not a standard file-allocation method.

Step 3 – note the trade-off: linked allocation avoids external fragmentation but makes direct (random) access slow, since reaching block k means following k pointers.

Final Answer: linked allocation $\Rightarrow$ B

Answer: (B) Go Back to Q34

Q35.

Solution

Concept – the natural join: The natural join $R \bowtie S$ pairs tuples that agree on all attributes common to R and S, and outputs each shared attribute only once.

Step 1 – match the described behaviour: "combine on common attributes, keep one copy of each shared attribute" is the definition of the natural join.

Step 2 – eliminate the others:

- Projection (π) selects columns from one relation.
- Selection (σ) filters rows of one relation by a predicate.
- Rename (ρ) only changes a relation's or attribute's name.

Step 3 – conclude: only the natural join combines two relations on matching common attributes.

Final Answer: natural join ($\bowtie$) $\Rightarrow$ A

Answer: (A) Go Back to Q35

Q36.

Solution

Concept – second normal form (2NF): A relation is in 2NF if it is in 1NF and no non-prime attribute depends on only *part* of a candidate key (no partial dependency). A non-prime attribute is one that is not part of any candidate key.

Step 1 – start from 1NF: all attribute values are already atomic.

Step 2 – add the 2NF restriction: eliminate every partial dependency, i.e. each non-prime attribute must depend on the *whole* candidate key, not a proper subset of it.

Step 3 – eliminate the others:

- "every determinant is a candidate key" is BCNF.
- "no transitive dependency" is the 3NF condition.
- "atomic values" is only the 1NF condition.

Final Answer: no non-prime attribute is partially dependent on a candidate key $\Rightarrow$ C

Answer: (C) Go Back to Q36

Q37.

Solution

Concept – atomicity (the "A" of ACID): Atomicity treats a transaction as all-or-nothing: if any part fails, the whole transaction is rolled back so that none of its effects remain.

Step 1 – match the described guarantee: "either all operations take effect or none do" is exactly atomicity.

Step 2 – distinguish the other properties:

- Consistency: a transaction moves the database from one valid state to another.
- Isolation: concurrent transactions do not interfere with one another.
- Durability: committed changes survive later crashes.

Step 3 – conclude: the all-or-nothing property is atomicity.

Final Answer: atomicity $\Rightarrow$ B

Answer: (B) Go Back to Q37

Q38.

Solution

Concept – removing duplicates in SQL: The `DISTINCT` keyword, placed right after `SELECT`, discards duplicate rows so that each distinct combination of the selected columns appears only once.

Step 1 – recall the syntax: `SELECT DISTINCT col FROM table;` returns each value of `col` once.

Step 2 – eliminate the others:

- `UNIQUE` is a table constraint, not a row-deduplication keyword in a `SELECT`.
- `GROUP BY` groups rows for aggregation; its main purpose is not simple deduplication.
- `HAVING` filters groups after aggregation.

Step 3 – conclude: the keyword that removes duplicate rows from a query result is `DISTINCT`.

Final Answer: `DISTINCT` $\Rightarrow$ A

Answer: (A) Go Back to Q38

Q39.

Solution

Concept – OSI layer of a switch (bridge): A switch forwards frames using their destination *MAC* addresses. MAC addressing lives in the data-link layer, so a switch is a layer-2 device.

Step 1 – identify the address it uses: the device S decides where to forward using MAC (physical/hardware) addresses.

Step 2 – locate MAC addressing in the OSI model: MAC is the lower sublayer of the data-link layer (layer 2).

Step 3 – eliminate the others:

- Layer 3 (network) is where routers work, using IP addresses.
- Layer 1 (physical) is where hubs/repeaters work, on raw bits.
- Layer 4 (transport) handles end-to-end connections, not frame forwarding.

Final Answer: data-link layer (layer 2) $\Rightarrow$ A

Answer: (A) Go Back to Q39

Q40.

Solution

Concept – CRC check-bit count: If the generator polynomial has degree r, then r check bits (the frame check sequence) are appended. For a generator written as a bit string of length L, the degree is $r = L - 1$.

Step 1 – read the generator: the bit pattern is 1011, which has 4 bits.

Step 2 – find its degree: degree $r =$ (number of bits) $- 1 = 4 - 1 = 3$.

Step 3 – number of appended check bits: equals the degree $= 3$.

Step 4 – cross-check: the leftmost 1 corresponds to x^3, confirming degree 3 and 3 CRC bits.

Final Answer: 3 check bits $\Rightarrow$ B

Answer: (B) Go Back to Q40

Q41.

Solution

Concept – Selective-Repeat window size: In Selective Repeat, the sender and receiver windows must each be at most half of the sequence-number space to avoid confusing new frames with retransmitted ones.

Step 1 – size of the sequence-number space: an n-bit field gives 2^n distinct numbers.

Step 2 – apply the half rule: maximum window $= \dfrac{2^n}{2} = 2^{n-1}$.

Step 3 – contrast with Go-Back-N: Go-Back-N allows a window up to $2^n - 1$, but Selective Repeat is limited to 2^{n-1}, which is stricter.

Final Answer: $2^{n-1} \Rightarrow$ C

Answer: (C) Go Back to Q41

Q42.

Solution

Concept – bandwidth–delay product: The bandwidth–delay product is (bandwidth in bits per second) $\times$ (one-way delay in seconds); it is the number of bits that fit on the link at once.

Step 1 – convert the bandwidth to bits per second: 1 Mbps $= 1 \times 10^6$ bits/s.

Step 2 – convert the delay to seconds: 10 ms $= 10 \times 10^{-3} = 10^{-2}$ s.

Step 3 – multiply: $(1 \times 10^6) \times (10^{-2}) = 10^{6-2} = 10^4$

Step 4 – state the result: $10^4 = 10{,}000$ bits.

Final Answer: 10,000 bits $\Rightarrow$ A

Answer: (A) Go Back to Q42

Q43.

Solution

Concept – number of subnets from borrowed bits: Extending the prefix from $/p_1$ to $/p_2$ borrows $(p_2 - p_1)$ host bits, producing $2^{(p_2-p_1)}$ subnets.

Step 1 – find the borrowed bits: from $/24$ to $/28$: $28 - 24 = 4$ bits.

Step 2 – raise 2 to that power: $2^4 = 16$

Step 3 – state the number of subnets: 16 equal-sized subnets.

Why the other options are wrong:

- A, 8: would be borrowing only 3 bits $(/27)$.
- D, 14: confuses subnet count with a usable-host count.

Final Answer: 16 subnets $\Rightarrow$ B

Answer: (B) Go Back to Q43

Q44.

Solution

Concept – Address Resolution Protocol (ARP): ARP maps a known IPv4 address to the corresponding MAC (hardware) address on the same local network, by broadcasting a request and receiving the owner's reply.

Step 1 – match the direction of the mapping: we know the IP address and want the MAC address, i.e. IP $\rightarrow$ MAC.

Step 2 – eliminate the others:

- DNS maps a domain name to an IP address.
- DHCP hands out IP addresses to hosts.
- RARP maps a MAC address to an IP address (the reverse direction).

Step 3 – conclude: the IP-to-MAC mapping is done by ARP.

Final Answer: ARP $\Rightarrow$ C

Answer: (C) Go Back to Q44

Q45.

Solution

Concept – IMAP vs POP3: IMAP (Internet Message Access Protocol) lets a client read and manage mail while the messages stay on the server, so multiple devices see the same synchronised mailbox. POP3 typically downloads and removes mail from the server.

Step 1 – match "keep on server and synchronise across devices": this is the defining feature of IMAP.

Step 2 – eliminate the others:

- SMTP is for *sending* mail, not retrieving it.
- POP3 usually downloads then deletes, so it does not keep the mailbox synchronised across devices.
- FTP is a file-transfer protocol, unrelated to e-mail retrieval.

Step 3 – conclude: the retrieve-and-keep-synchronised protocol is IMAP.

Final Answer: IMAP $\Rightarrow$ B

Answer: (B) Go Back to Q45

Q46.

Solution

Concept – DNS transport and port: For ordinary (short) queries, DNS uses UDP for speed and low overhead, on the well-known port 53. (It switches to TCP for large responses or zone transfers.)

Step 1 – choose the transport protocol: short request/response lookups favour connectionless UDP.

Step 2 – recall the well-known port: DNS uses port 53.

Step 3 – eliminate the others:

- TCP port 53 is used only for large transfers/zone transfers, not ordinary lookups.
- Port 80 is HTTP; port 25 is SMTP, both unrelated to DNS.

Final Answer: UDP, port 53 $\Rightarrow$ B

Answer: (B) Go Back to Q46

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	B	2	C	3	A	4	C	5	A
6	D	7	A	8	B	9	D	10	D
11	A	12	B	13	D	14	A	15	D
16	D	17	B	18	D	19	D	20	C
21	B	22	A	23	D	24	D	25	A
26	B	27	C	28	C	29	A	30	C
31	C	32	D	33	C	34	B	35	A
36	C	37	B	38	A	39	A	40	B
41	C	42	A	43	B	44	C	45	B
46	B								