Write answers to all questions(If duplicate occur then skip) neatly by hand on A4-size pages and submit them through the file-attachment channel or spiral binding , including a front page, by 2:00 PM on 19 January 2026.

MCQ Questions

Q1. A problem where the output is only YES/NO is best classified as:
A) Optimization problem
B) Decision problem
C) Search problem
D) Enumeration problem
Answer: B

Q2. A problem instance is:
A) A general form of a problem
B) A specific input case of a problem
C) A programming language feature
D) A data structure
Answer: B

Q3. Requirement: "Create a program to calculate a student's final grade." Which clarification question is most essential?
A) Which IDE should be used?
B) What is the student's name?
C) What are the input components and their weightage (marks/percentage)?
D) Should the output be in color?
Answer: C

Q4. For "Compute average of N numbers", a valid pre-condition is:
A) Output must be printed
B) $N > 0$
C) Average must be integer
D) Numbers must be sorted
Answer: B

Q5. Algorithm correctness means:
A) Uses minimum memory always
B) Produces correct output for all valid inputs
C) Runs fastest among all algorithms
D) Uses recursion
Answer: B

Q6. Which algorithm is asymptotically more efficient for large n?
A) $O(n^2)$
B) $O(n \log n)$
C) $O(n^3)$
D) $O(2^n)$
Answer: B

Q7. Role of data structures in problem solving is mainly to:
A) Increase program file size
B) Organize data for efficient access and updates
C) Replace algorithms
D) Avoid input/output
Answer: B

Q8. A program reads age. Which is the best input validation condition?
A) age != 0
B) age >= 0 && age <= 120
C) age % 2 == 0
D) age < 1000 only
Answer: B

Q9. ASCII value of character 'A' is:
A) 97
B) 65
C) 48
D) 66
Answer: B

Q10. Which loop is exit-controlled?
A) for
B) while
C) do-while
D) none
Answer: C

Q11. Which is best for sentinel-controlled repetition (unknown count, stop at -1)?
A) for(i=0;i<n;i++)
B) while(x!=-1)
C) switch(x)
D) if(x==-1)
Answer: B

Q12. What is the output?

```
int i,j,sum=0;
for(i=1;i<=3;i++){
  for(j=1;j<=i;j++) sum += j;
}
printf("%d", sum);
```

A) 6
B) 10
C) 12
D) 9
Answer: B

Q13. Which C statement correctly reads an integer x?
A) input(x);
B) read x;
C) scanf("%d",&x);

D) cin >> x;
Answer: C

Q14. C is mainly a:
A) Typeless language
B) Strongly typed language
C) Markup language
D) Query language
Answer: B

Q15. For x = -2, what prints?

```
if (x > 0)
  printf("P");
  printf("Q");
```

A) P only
B) Q only
C) PQ
D) Nothing
Answer: B (because Q is outside the if-block)

Q16. In 8-bit 2's complement, adding 01111111 (127) and 00000001 (1) results in:
A) 10000000 and overflow occurs
B) 10000000 and no overflow
C) 00000000 and overflow occurs
D) 11111111 and no overflow
Answer: A

Q17. In IEEE 754 single precision, the sign bit indicates:
A) Mantissa length
B) Positive/negative number
C) Exponent value
D) ASCII code
Answer: B

Q18. What is the output?

```
int a=5;
int b = a++ + ++a;
printf("%d %d", a, b);
```

A) 6 11
B) 7 12
C) 7 11
D) 6 12
Answer: B

Q19. Incremental compilation/testing means:
A) Write full code then test once
B) Compile and test small parts step-by-step as you build
C) Only test with large inputs
D) Avoid debugging tools
Answer: B

Q20. If AP has first term a=2 and common difference d=3, the 5th term is:
A) 11
B) 14
C) 17
D) 20
Answer: B ($a_5 = 2 + 4\times3 = 14$)

Q21. To test whether n is prime efficiently (basic method), we should check divisors up to:
A) n
B) n/2
C) √n
D) 2n
Answer: C

Q22. To extract digits left to right (e.g., 543 → 5,4,3) without reversing, you must first:
A) Use %10 repeatedly
B) Find the highest power of 10 ≤ n
C) Convert to binary
D) Use switch-case
Answer: B

Q23. What is the output?

```
int n=121, rev=0, t=n;
while(t>0){ rev=rev*10 + t%10; t/=10; }
printf("%s", (rev==n) ? "PAL" : "NOT");
```

A) PAL
B) NOT
C) 121
D) 0
Answer: A

Q24. continue in a loop:
A) terminates the loop immediately
B) skips remaining statements of current iteration
C) exits the program
D) repeats the previous iteration
Answer: B

Q25. In base conversion (decimal to binary), repeatedly divide by 2 and collect:
A) Quotients
B) Remainders
C) Products
D) Averages
Answer: B

Q26. What is the output?

```
int x=2;
switch(x){
  case 1: printf("A");
  case 2: printf("B");
  case 3: printf("C"); break;
```

```
  default: printf("D");
}
```

A) B
B) BC
C) ABC
D) BD
Answer: B

Q27. A number is Armstrong (3-digit) if it equals:
A) sum of its digits
B) product of its digits
C) sum of cubes of its digits
D) sum of squares of its digits
Answer: C

Q28. What is the main issue in this sentinel loop?

```
int x, sum=0, count=0;
while(x!=-1){
  scanf("%d",&x);
  sum += x;
  count++;
}
```

A) scanf cannot be used in loops
B) x is uninitialized and sentinel (-1) is included in sum/count
C) sum must be float
D) while loop never executes
Answer: B

Q29. else-if ladder is used when:
A) only one condition exists
B) multiple mutually exclusive conditions exist
C) only loops are needed
D) arrays are required
Answer: B

Q30. A palindrome number means:
A) divisible by 2
B) reads same forward and backward
C) sum of digits is prime
D) has even number of digits
Answer: B

Q31. A function prototype is mainly used to:
A) allocate memory to function
B) inform compiler about function’s return type and parameters
C) stop recursion
D) print function output
Answer: B

Q32. What will happen with this code?

```
int fact(int n){
  return n * fact(n-1);
}
```

A) Correct factorial
B) Compile-time error always
C) Infinite recursion / stack overflow (no base case)
D) Returns 0 always
Answer: C

Q33. Binary search requires the array to be:
A) reversed
B) sorted
C) of even size
D) 2D array
Answer: B

Q34. In bubble sort, number of swaps for array [4,3,2,1] is:
A) 3
B) 4
C) 6
D) 9
Answer: C

Q35. Correct way to swap using pointers in C is:
A) swap(a,b)
B) void swap(int *x,int *y){int t=*x;*x=*y;*y=t;}
C) void swap(int x,int y){x=y;}
D) swap(&x,&y,&t)
Answer: B

Q36. What is wrong here?

```
int a[5]={1,2,3,4,5};
for(int i=0;i<=5;i++) printf("%d ", a[i]);
```

A) Array cannot be printed in loop
B) Off-by-one: accesses a[5] out of bounds
C) printf needs %f
D) i must start from 1
Answer: B

Q37. strlen("CSE") returns:
A) 2
B) 3
C) 4
D) 0
Answer: B

Q38. What is the output?

```
int a=2,b=3,c=4;
a = b = c;
printf("%d %d %d", a,b,c);
```

A) 2 3 4
B) 4 3 4
C) 4 4 4
D) 3 4 4
Answer: C

Q39. Matrix multiplication A(m×n) × B(p×q) is possible only if:
A) m=q
B) n=p
C) m=p
D) n=q
Answer: B

Q40. A program crashes due to segmentation fault. Best debugging action is:
A) Add more printf everywhere only
B) Use debugger breakpoints + watch variables / check pointer values
C) Rewrite in another language
D) Delete all functions
Answer: B

Q41. What is the output?

```
int i,j,c=0;
for(i=1;i<=4;i++){
  for(j=1;j<=4;j++){
    if(j==i) continue;
    if(i+j==6) break;
    c++;
  }
}
printf("%d", c);
```

A) 8 B) 9 C) 10 D) 11
Ans: B

Q42. What is the output?

```
int x=0,y=1;
if(x)
  if(y) printf("A");
  else printf("B");
printf("C");
```

A) AC B) BC C) C D) ABC
Ans: C

Q43. What is the output?

```
int n=2;
switch(n){
  case 1: printf("1");
  case 2: printf("2");
  case 3: printf("3"); break;
  default: printf("D");
```

```
}
```

A) 2 B) 23 C) 123 D) 2D
Ans: B

Q44. What is the output?

```
int a=0,b=0;
if(a++ && ++b) { }
printf("%d %d", a,b);
```

A) 0 1 B) 1 1 C) 1 0 D) 0 0
Ans: C

Q45. What is the output?

```
int i=0;
while(++i<5){
  if(i%2==0) continue;
  printf("%d", i);
}
```

A) 1234 B) 13 C) 24 D) 135
Ans: B

Q46. What is the output?

```
int i, s=0;
for(i=1; i<=4; i++, s+=i) ;
printf("%d", s);
```

A) 10 B) 14 C) 15 D) 9
Ans: B

Q47. What is the output?

```
int a[]={1,2,3,4,5};
int i=0, sum=0;
while(i<5){
  sum += a[i];
  i += a[i]%2;
  i++;
}
printf("%d", sum);
```

A) 6 B) 7 C) 8 D) 9
Ans: D

Q48. What is the output?

```
int f(int n){
  if(n<=1) return 1;
  return n * f(n-2);
```

```
}
int main(){
  printf("%d", f(6));
}
```

A) 36 B) 48 C) 72 D) 120
Ans: B

Q49. What is the output?

```
char s[]="CSE";
printf("%zu %zu", sizeof(s), strlen(s));
```

A) 3 3 B) 4 4 C) 4 3 D) 3 4
Ans: C

Q50. What is the output?

```
int a=1,b=1,c=1;
int r = a++ && ++b || c++;
printf("%d %d %d %d", r,a,b,c);
```

A) 1 2 2 2 B) 1 2 2 1 C) 0 2 2 2 D) 1 1 2 1
Ans: B

5 MARK QUESTIONS

Q1. Convert $(345)_{10}$ into binary, octal, and hexadecimal. Show all intermediate steps.

Q2. Convert $(101101.011)_2$ into decimal and hexadecimal.

Q3. Convert $(7A3)_{16}$ into binary and decimal.

Q4. Convert $(753)_8$ into binary and decimal.

Q5. Represent −27 in 8-bit signed magnitude, 1's complement, and 2's complement. Compare the three representations.

Q6. Find the range of values representable using 8-bit signed magnitude and 8-bit 2's complement. Write both ranges clearly.

Q7. Perform binary addition using 8-bit 2's complement: (+75) + (+60). State whether overflow occurs and justify.

Q8. Perform subtraction using 2's complement: (−35) + (+18) in 8-bit representation. Show steps.

Q9. Explain why 2's complement is preferred over 1's complement for signed integer arithmetic (any 3 valid reasons).

Q10. Convert the decimal floating number $(-13.25)_{10}$ into IEEE 754 single precision (sign, exponent, mantissa).

Q11. Given an IEEE 754 single-precision pattern:
0 10000010 01000000000000000000000
Find the decimal value and explain the steps.

Q12. Convert the character string "JIS" into ASCII codes (decimal). Also write the binary of each ASCII code.

Q13. Differentiate ASCII and UNICODE with examples and mention why Unicode is required in modern computing.

Q14. Convert decimal number 156 into base-2, base-8, base-16 using repeated division method.

Q15. Design an algorithm (pseudocode) to convert a number from base-b to decimal, where $2 \leq b \leq 16$.

Q16. In 8-bit 2's complement, show the effect of overflow for: (+127) + (+5). Explain overflow condition in signed addition.

Q17. Using Taylor series, write the expression and compute sin(x) approximately up to 3 non-zero terms for x = 0.5 radians.

Q18. Using Taylor series, write the expression and compute an approximate value of π using any one known series up to 3 terms.

Q19. Write a C program to print the first N terms of Fibonacci series using a loop.

Q20. Write a C program to generate AP and GP for N terms given first term and difference/ratio.

Q21. Write a C program to display a right-angled triangle pattern of * of height N.

Q22. Write pseudocode + flowchart steps for finding the largest of three numbers using selection.

Q24. Write a C program to check whether a number is Palindrome.

Q25. Write a C program to check whether a number is Armstrong (for 3-digit numbers).

Q26. Write a C program to check whether a number is Prime and also print all primes up to N.

Q27. Write a C program to print prime factors of a given number.

Q28. Write a C program to check whether a number is Perfect number.

Q29. Write a C program to compute factorial using iteration.

Q30. Write a C program to convert a number from decimal to binary.

Q31. A sequence is read until sentinel -1. Write a C program to find max, min, sum, average ensuring the sentinel is not included.

Q32. Write a C program using functions to compute GCD of two numbers (function + main).

Q33. Write a recursive C program to compute factorial.

Q34. Write a C program to read an array of N elements and find max, min, sum, average.

Q35. Write a C program to perform binary search on a sorted array.

Q36. Write a C program to implement any one sorting algorithm (Bubble/Selection/Insertion) for an integer array.

Q37. Write a C program for matrix addition of two matrices.

Q38. Write a C program for matrix multiplication (A×B), include dimension validation.

Q40. Write a C program using string functions to count vowels, consonants, digits, spaces in a line of text.

Q41. Differentiate algorithm correctness and algorithm efficiency with one example.

Q42. Explain the difference between a problem, problem instance, and solution with a suitable example.

Q43. Requirement: "Build a program to compute electricity bill."
Write the problem specification (inputs, outputs, constraints) and list 5 clarification questions.

Q44. Compare for, while, do-while loops based on entry/exit control and best use-case (any 3 points).

Q45. Compare syntax error, runtime error, logical error with one C example for each and how to detect them.

Q46. Compare Signed Magnitude vs 1's Complement vs 2's Complement (range, zero representation, arithmetic simplicity).

Q47. Compare if-else ladder vs switch-case (limitations, readability, performance, suitable scenarios).

Q48. Compare Linear Search vs Binary Search (precondition, complexity, best case/worst case, when to use).

Q49. Compare Recursion vs Iteration (memory, speed, ease, termination issues) with one example problem.

Q50. Explain a systematic debugging approach (breakpoints, watch, step-in/over, test cases) for finding a wrong output in a C program.

Q51. A household wants a C program to generate an electricity bill. The user enters Customer ID, Name, and Units Consumed. Billing rules: 0–100 units @ ₹3/unit, 101–200 @ ₹4/unit, 201–300 @ ₹5/unit, and >300 @ ₹6/unit; add a fixed charge ₹50. The program must print total amount. Assume units are monthly. Validate: ID/name non-empty; units must be integer ≥ 0.

Q52. Differentiate between algorithm efficiency and algorithm correctness with suitable examples.

List of C Programs & Previous year Question

5 marks each question

1. Sum of first N natural numbers
   Write a C program to input a positive integer n and find the sum 1 + 2 + … + n using a loop.
2. Sum of even numbers up to N
   Write a C program to input n and find the sum of all even numbers from 1 to n using a loop.
3. Factorial of a number
   Write a C program to input an integer n and find n! using a loop (without recursion).
4. Table of a number
   Write a C program to read an integer and print its multiplication table from 1 to 10 using a loop.
5. Reverse counting
   Write a C program to print numbers from n down to 1 using a while loop.
6. Digits count
   Write a C program to count the number of digits in a given integer using a loop.
7. Sum of digits
   Write a C program to find the sum of digits of a given number using a loop.
8. Reverse a number
   Write a C program to reverse a given integer (e.g., 1234 → 4321) using a loop.
9. Palindrome number check
   Using a loop, write a C program to check whether a given integer is a palindrome or not.
10. Armstrong number check (3-digit)
    Write a C program using loops to check whether a 3-digit number is an Armstrong number (sum of cubes of digits equals the number).
11. Power using repeated multiplication
    Write a C program to compute a^b (a to the power b) using a loop (without using pow()).
12. Print all even numbers between two numbers
    Write a C program to input two integers low and high and print all even numbers between them using a loop.
13. Print all odd numbers between two numbers
    Similar to above, but print all odd numbers between low and high using a loop.
14. Prime number check
    Write a C program to check whether a given number is prime or not using a loop.
15. Print all primes in a range
    Write a C program that uses loops to print all prime numbers between 1 and n.
16. Fibonacci series (first N terms)
    Write a C program to print the first n Fibonacci numbers using a loop.
17. Fibonacci series up to a limit
    Write a C program to print all Fibonacci numbers less than or equal to a given number n using a loop.
18. LCM using repeated addition / loop
    Write a C program to find the LCM of two numbers using loops (by checking multiples).
19. GCD using repeated subtraction / loop
    Write a C program to find the GCD of two numbers using loops (Euclidean algorithm with while).
20. Menu-driven calculator using loop
    Write a C program that repeatedly asks the user to choose an operation (+, -, *, /) and two numbers, performs the operation, and continues until the user chooses to exit (use a loop).
21. Sum of series: 1 + 1/2 + 1/3 + … + 1/n
    Write a C program using a loop to compute the sum of the harmonic series up to n terms (use float or double).
22. Print characters A–Z using loop
    Write a C program to print all uppercase letters from ‘A’ to ‘Z’ using a loop.
23. Count vowels in a string (using loop)
    Write a C program to input a string and count the number of vowels using a loop (no string library iteration functions).

24. Count spaces and digits in a string
    Write a C program that reads a line of text and uses a loop to count spaces, digits, and other characters.
25. Sum of elements in an array using loop
    Write a C program to input n elements in an array and find their sum using a loop.
26. Maximum element in an array (loop)
    Write a C program to read n integers into an array and find the maximum value using a loop.
27. Linear search using loop
    Write a C program to perform linear search in an array using a loop and indicate whether the element is found or not.
28. Frequency of a given element in an array
    Write a C program to count how many times a particular element appears in an array using a loop.
29. Reverse an array using loop
    Write a C program that reads an array of n integers and prints the array elements in reverse order using a loop.
30. Bubble sort using nested loops
    Write a C program to sort an array of integers using the bubble sort algorithm (nested loops).
31. Pattern 1: Right-angled triangle of stars
    For input n, print the following pattern using nested loops:
    *
    * *
    * * *
    … (n rows)
32. Pattern 2: Inverted right-angled triangle
    For input n, print the pattern using nested loops:
    * * * *
    * * *
    * *
    *
33. Pattern 3: Floyd's triangle (1, 2, 3, …)
    For input n, print Floyd's triangle using nested loops:
    1
    2 3
    4 5 6
    … Up to N terms
34. Pattern 4: Number triangle
    For input n, print:
    1
    1 2
    1 2 3
    … up to n rows using loops.
35. Pattern 5: Pyramid of stars
    For input n, print a pyramid (center-aligned) using nested loops, e.g., for n=3:
    *
    * *
    * * *
36. Sum of even and odd elements separately
    Write a C program to read n integers and use a loop to compute the sum of even elements and the sum of odd elements separately.
37. Count positive, negative, zero numbers
    Read n integers and use a loop to count how many are positive, negative, and zero.
38. Read until -1 and compute average
    Write a C program that repeatedly reads integers from the user until -1 is entered, then prints the average of all numbers (excluding -1) using a loop.

39. Guessing game with limited attempts
    Write a C program that generates / assumes a secret number and allows the user at most 5 attempts to guess it, using a loop.
40. Check if a number is perfect
    A number is perfect if the sum of its proper divisors equals the number (e.g., 6 = 1+2+3). Write a C program using a loop to check if a number is perfect.
41. Print all perfect numbers in a range
    Using loops (nested), print all perfect numbers between 1 and n.
42. Count the occurrence of each digit in a number
    Write a C program that uses loops to count how many times each digit (0–9) appears in a given integer.
43. Convert decimal to binary (using loop)
    Write a C program using loops to convert a decimal number to its binary representation and print it.
44. Print digits of a number in words
    Using loops and switch-case, write a C program that prints each digit of a number in words (e.g., 120 → ONE TWO ZERO).
45. Check if a number is strong number
    A strong number is one whose sum of factorial of digits equals the number (e.g., 145 = 1! + 4! + 5!). Write a C program using loops to check this.
46. Sum of series: $1^2 + 2^2 + 3^2 + \ldots + n^2$
    Write a C program to find the sum of squares of first n natural numbers using a loop.
47. Sum of series: $1^3 + 2^3 + 3^3 + \ldots + n^3$
    Write a C program to find the sum of cubes of first n natural numbers using a loop.
48. Print all factors of a number
    Write a C program using a loop to print all the factors of a given number.
49. Check if a number is composite
    Write a C program using loops to check if a given number is composite (non-prime and > 1) and print its smallest factor (other than 1).
50. Do-while menu for repeated input
    Write a C program using a do-while loop to repeatedly display a menu with options (e.g., "1. Enter number 2. Print square 3. Exit") and perform the selected operation until the user chooses to exit
51. Write a C program to print the following right-angled triangle of numbers for a given n:

    ```
    1
    1 2
    1 2 3
    1 2 3 4
    ------ n terms
    ```

52. Write a C program to print the following pattern of repeated row numbers for a given n:

    ```
    1
    2 2
    3 3 3
    4 4 4 4
    ------- n terms
    ```

53. Write a C program to print the following pyramid pattern of stars (centre aligned) for a given n:

    ```
      *
     * *
    ```

```
 * * *
* * * *
```

54. Write a C program to print the following inverted pyramid pattern of stars for a given n:

```
* * * *
 * * *
  * *
   *
```

55. Write a C program to print the following diamond pattern of stars for a given odd number n (height):

```
   *
  * *
 * * *
  * *
   *
```

56. Write a C program to read N integers into an array and print sum and average.
57. Write a C program to find maximum and minimum elements of an array.
58. Write a C program to find the second largest element in an array (without sorting).
59. Write a C program to reverse an array in-place.
60. Write a C program to count frequency of a given element in an array.
61. Write a C program for linear search and print the index if found, otherwise print “Not Found”.
62. Write a C program to perform binary search on a sorted array (include necessary validation).
63. Implement Bubble Sort for an integer array and print the sorted array.
64. Write a C program to perform matrix addition for two matrices of same order.
65. Write a C program to multiply two matrices and include a check for valid dimensions.
66. Write a function swap() to swap two integers using pointers (call by reference).
67. Write a C program to find the sum of array elements using pointer arithmetic (no indexing a[i]).
68. Explain with example: pointer and array relationship (arr, &arr[0], *(arr+i)).
69. Write a C program to find string length using pointers (do not use strlen).
70. Write a C program to copy one string to another using pointers (no strcpy).
71. Write a C program to reverse a string using pointers.
72. Write a function that returns the address of the maximum element in an array using pointers.
73. Write a program using pointer to pointer to update a variable value and print before/after values.
74. Write a C program using a function isPrime(n) to check prime and print result.
75. Write a C program using a function to compute GCD of two numbers (Euclid method).
76. Write a recursive function to compute factorial of a number.
77. Design a modular solution: write functions for readArray(), printArray(), sumArray() and call them from main().
78. Write a function to return maximum and minimum of an array (use parameters + return strategy of your choice).
79. Write a C program to perform binary search using a separate function binSearch(arr,n,key) returning index or -1.
80. Write a program to implement bubble sort using a function bubbleSort(arr,n) and explain how you test it with 3 test cases.
81. Write functions for matrix addition: readMatrix(), addMatrix(), printMatrix() for fixed order (say 3×3).
82. Explain the role of function prototype with an example where missing prototype causes a logical/compilation issue.

83. Given a program producing wrong output, write a short debugging plan (breakpoint, watch variables, step execution) to locate the error in a function-based program.
84. Write a C program that calculates the factorial of a given non-negative integer (n) using a recursive function. Include input validation (reject negative values) and display the result.
85. Write a C program that accepts three side lengths (a, b, c) from the user, first validates whether they can form a triangle (triangle inequality), then uses a switch–case to classify the triangle as Equilateral / Isosceles / Scalene. Finally, display the perimeter and the area of the triangle (use Heron's formula for area).
86. Convert $(+13.25)_{10}$ into IEEE 754 single precision (32-bit). Show sign, exponent, mantissa.
87. Convert $(-0.75)_{10}$ into IEEE 754 single precision (32-bit).
88. Convert $(+156.5)_{10}$ into IEEE 754 double precision (64-bit). Show all fields.
89. Convert $(-45.125)_{10}$ into IEEE 754 double precision (64-bit).
90. Decode this IEEE 754 single precision pattern and find the decimal value:
    0 10000010 01000000000000000000000
91. Decode this IEEE 754 double precision pattern and find the decimal value:
    1 10000000101 0010100000000000000000000000000000000000000000000000
92. Explain the meaning of bias in IEEE 754. Calculate the bias for single and double precision.
93. Compare single vs double precision in terms of bits for sign/exponent/fraction, approximate precision, and typical use cases.
94. For IEEE 754 single precision, analyze whether overflow/underflow may occur when storing very large or very small values. Give one example for each.
95. A program gives different results using float vs double for repeated additions (e.g., adding 0.1 many times). Explain why this happens in IEEE 754 and suggest a safe approach.
96. Calculate the value of $(x)_2$ and $(y)_8$ from the following equations:

    $70-x=(86)_{16}-(145)_8$

    $y+(1100100)_2+(12)_{16}=(123.75)_{10}$

97. Calculate the value of $(x)_2$ and $(y)_8$ from the following equations:

    $90-x=(6E)_{16}-(110)_8$

    $y+(1100000)_2+(36)_{16}=162.25$

98. Design a clear problem specification (inputs, outputs, constraints, validations) for: "Compute electricity bill for customers with slab rates."
99. Write pseudocode (structured constructs) for finding max, min, sum, average of numbers read until sentinel -1 (exclude -1).
100. Create a flowchart to generate the first N Fibonacci numbers using repetition and selection (handle $N \leq 0$ with validation).
101. Develop a C program using functions to check whether a number is prime and print primes in a range [a, b].
102. Create a C program to convert a decimal number to binary and hexadecimal without using library conversion functions.
103. Design a modular solution (top-down) and write C code for matrix multiplication, including dimension validation.
104. Create a C program to compute sin(x) using Taylor series up to N terms (use loop; validate N and x).
105. Create an algorithm + C program to find whether a number is Armstrong (generalized to k-digits).
106. Design a C program that reads an array and finds median and mode (assume N is odd for median; include constraints).

107. Create a debugging-oriented test plan (at least 6 test cases) for a palindrome-check program (include boundary/invalid cases).
108. Two approaches to find maximum of N numbers: (i) linear scan, (ii) sort then pick last. Evaluate which is better and justify using time complexity.
109. A requirement says: “Find average of N numbers.” Evaluate the requirement for ambiguity and write 5 missing details that must be clarified.
110. Given a loop-based solution for factorial and a recursive solution, evaluate which is preferable in C for large n and justify (stack/memory/limits).
111. Evaluate the choice between for, while, and do-while for sentinel-controlled input problems. Justify with one scenario.
112. In 8-bit signed arithmetic, evaluate whether overflow occurs for (+100) + (+60) in 2’s complement, and justify using sign/overflow rules.
113. Evaluate why 2’s complement is preferred over 1’s complement for arithmetic operations (give 3 technical reasons).
114. Given a program using switch and if-else ladder for the same menu problem, evaluate which is more suitable and justify based on constraints/clarity.
115. For searching in an array, evaluate when linear search is better than binary search (give 3 conditions).
116. What is the meaning of header file in C? Describe 3 header file briefly with C program.