**Pre-increment & Post-increment Questions**

1. **What will be the output of the following code?**

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

   - a) 5 6
   - b) 6 6
   - c) 6 5
   - d) 5 5

2. **Predict the output:**

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

   - a) 5 6
   - b) 6 6
   - c) 6 5
   - d) 5 5

3. **Determine the output of the following:**

```
int x = 10;
printf("%d", x++ + ++x);
```

   - a) 21
   - b) 20
   - c) 19
   - d) Undefined behavior

4. **What will be the value of x and y after the following code executes?**

```
int x = 2, y;
y = x++ + ++x;
```

   - a) x = 4, y = 6
   - b) x = 3, y = 5
   - c) x = 4, y = 5
   - d) x = 3, y = 6

5. **Predict the output:**

```
int a = 10;
int b = 20;
int c = a++ + --b;
printf("%d %d %d", a, b, c);
```

- a) 11 19 30
- b) 10 20 30
- c) 11 19 29
- d) 10 19 29

**Logical Operators and Expressions**

6. **What will be the output of this code?**

```
int a = 5, b = 10;
printf("%d", (a++ > 5) && (b++ > 10));
```

- a) 0
- b) 1
- c) 5
- d) 10

7. **Find the result of the following logical expression:**

```
int x = 5, y = 10;
printf("%d", (x++ > 4) || (++y > 10));
```

- a) 0
- b) 1
- c) 5
- d) 10

8. **Evaluate the output of:**

```
int a = 2;
int b = 3;
int result = (++a * b++) / --b;
printf("%d", result);
```

- a) 6
- b) 7

- c) 8
- d) 9

**Nested Increments and Mixed Operations**

9. **Predict the output:**

```
int x = 1;
x = ++x + x++ + x;
printf("%d", x);
```

- a) 5
- b) 6
- c) 7
- d) 8

10. **Find the output:**

```
int a = 5, b = 10;
int result = (++a * b--) + (b++ + --a);
printf("%d", result);
```

- a) 85
- b) 80
- c) 75
- d) 70

**Conditional Increments**

11. **What will be the output of the following program?**

```
int a = 5, b = 10;
if (a++ > 5)
   b--;
else
   ++b;
printf("%d %d", a, b);
```

- a) 5 10
- b) 6 9
- c) 6 11
- d) 5 11

12. **Evaluate the final value of c:**

```
int a = 10, b = 20;
int c = (a++ > b) ? ++a : b--;
```

- a) 10
- b) 11
- c) 20
- d) 19

13. **Predict the output of this code:**

```
int a = 3;
printf("%d", a++ + ++a);
```

- a) 6
- b) 7
- c) 8
- d) 9

14. **Evaluate the final values of a and b:**

```
int a = 2, b;
b = a++ + ++a + a++;
printf("%d %d", a, b);
```

- a) 5 8
- b) 4 7
- c) 4 8
- d) 3 7

15. **What is the output of the following?**

```
int x = 1, y = 5;
int z = x++ + y++ + ++x;
printf("%d", z);
```

- a) 8
- b) 9
- c) 10
- d) 11

---

**2. Conditional (Ternary) Operators**

16. **Find the output of this code:**

```
int a = 3, b = 5;
int max = (a > b) ? a : b;
printf("%d", max);
```

- a) 3
- b) 5
- c) 0
- d) 8

17. **Predict the output:**

```
int a = 10, b = 20;
int c = (a > b) ? a : b;
printf("%d", c);
```

- a) 10
- b) 20
- c) 0
- d) 30

18. **What is the result of the following code?**

```
int x = 5;
int result = (x % 2 == 0) ? x + 2 : x + 1;
printf("%d", result);
```

- a) 6
- b) 7
- c) 8
- d) 5

---

**3. Bitwise Operators**

19. **Evaluate the output of this bitwise operation:**

```
int a = 5;  // Binary: 0101
int b = 3;  // Binary: 0011
printf("%d", a & b);
```

- a) 1
- b) 2
- c) 3
- d) 5

20. **Find the result of the following bitwise shift:**

```
int x = 4;  // Binary: 0100
printf("%d", x << 1);
```

- a) 2
- b) 4
- c) 8
- d) 16

21. **What will the output be?**

```
int x = 6;  // Binary: 0110
printf("%d", x ^ 3);  // XOR with 0011
```

- a) 5
- b) 2
- c) 1
- d) 4

---

**4. Loops**

22. **What will be the output of the following for loop?**

```
for (int i = 0; i < 5; i++) {
   printf("%d ", i);
}
```

- a) 0 1 2 3 4
- b) 1 2 3 4 5
- c) 0 1 2 3 4 5
- d) 1 2 3 4

23. **How many times will this while loop execute?**

```
int i = 0;
while (i < 3) {
```

```
    printf("%d ", i);
    i++;
}
```

- a) 2
- b) 3
- c) 4
- d) Infinite

24. **Predict the output of this do-while loop:**

```
int i = 0;
do {
    printf("%d ", i);
    i++;
} while (i < 3);
```

- a) 0 1 2
- b) 0 1 2 3
- c) 1 2 3
- d) 0 1 2 3 4

---

**5. Functions**

25. **What will be the output when this function is called?**

```
int add(int a, int b) {
    return a + b;
}
int main() {
    int result = add(3, 4);
    printf("%d", result);
}
```

- a) 3
- b) 4
- c) 7
- d) 8

26. **Find the output of this recursive function:**

```
int factorial(int n) {
   if (n == 0)
      return 1;
   else
      return n * factorial(n - 1);
}
int main() {
   printf("%d", factorial(3));
}
```

- a) 6
- b) 9
- c) 5
- d) 3

---

**6. Arrays and Pointers**

27. **What will be the output of the following code?**

```
int arr[] = {1, 2, 3, 4};
printf("%d", arr[2]);
```

- a) 1
- b) 2
- c) 3
- d) 4

28. **Evaluate the result when using pointers:**

```
int a = 10;
int *p = &a;
printf("%d", *p);
```

- a) 10
- b) 11
- c) Address of a

- d) 0

---

**7. Star Pattern Questions**

**1. Simple Right-Angled Triangle Pattern**

**Problem:** Write a program to print the following pattern:

markdown

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

**Code:**

```
#include <stdio.h>


int main() {
   int i, j, n = 5;
   for (i = 1; i <= n; i++) {
      for (j = 1; j <= i; j++) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

**2. Inverted Right-Angled Triangle Pattern**

**Problem:** Write a program to print the following inverted triangle:

markdown

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

*

**Code:**

```
#include <stdio.h>

int main() {
   int i, j, n = 5;
   for (i = n; i >= 1; i--) {
      for (j = 1; j <= i; j++) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

**3. Full Pyramid Pattern**

**Problem:** Write a program to print the following full pyramid:

markdown

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

**Code:**

```
#include <stdio.h>

int main() {
   int i, j, space, n = 5;
   for (i = 1; i <= n; i++) {
      for (space = 1; space <= n - i; space++) {
         printf(" ");
      }
```

```
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

**4. Inverted Full Pyramid**

**Problem:** Write a program to print the following inverted full pyramid:

markdown

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

**Code:**

```
#include <stdio.h>

int main() {
    int i, j, space, n = 5;
    for (i = n; i >= 1; i--) {
        for (space = 1; space <= n - i; space++) {
            printf(" ");
        }
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

**5. Diamond Pattern**

**Problem:** Write a program to print the following diamond shape:

markdown

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

**Code:**

```
#include <stdio.h>

int main() {
    int i, j, space, n = 5;

    // Upper half of the diamond
    for (i = 1; i <= n; i++) {
        for (space = 1; space <= n - i; space++) {
            printf(" ");
        }
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }

    // Lower half of the diamond
    for (i = n - 1; i >= 1; i--) {
```

```
        for (space = 1; space <= n - i; space++) {
            printf(" ");
        }
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

**6. Hollow Square Pattern**

**Problem:** Write a program to print the following hollow square pattern:

markdown

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

**Code:**

```
#include <stdio.h>

int main() {
    int i, j, n = 5;

    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            if (i == 1 || i == n || j == 1 || j == n)
                printf("* ");
            else
```

```
            printf("  ");
        }
        printf("\n");
    }

    return 0;
}
```

**7. Hollow Diamond Pattern**

**Problem:** Write a program to print the following hollow diamond pattern:

markdown

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

**Code:**

```
#include <stdio.h>

int main() {
    int i, j, space, n = 5;

    // Upper half of the hollow diamond
    for (i = 1; i <= n; i++) {
        for (space = 1; space <= n - i; space++) {
            printf(" ");
        }
        for (j = 1; j <= 2 * i - 1; j++) {
```

```
            if (j == 1 || j == 2 * i - 1)
                printf("*");
            else
                printf(" ");
        }
        printf("\n");
    }

    // Lower half of the hollow diamond
    for (i = n - 1; i >= 1; i--) {
        for (space = 1; space <= n - i; space++) {
            printf(" ");
        }
        for (j = 1; j <= 2 * i - 1; j++) {
            if (j == 1 || j == 2 * i - 1)
                printf("*");
            else
                printf(" ");
        }
        printf("\n");
    }

    return 0;
}
```

**8. Right-angled Number Pyramid**

**Problem:** Write a program to print the following right-angled number pyramid:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

**Code:**

```
#include <stdio.h>


int main() {
   int i, j, n = 5;
   for (i = 1; i <= n; i++) {
      for (j = 1; j <= i; j++) {
         printf("%d ", j);
      }
      printf("\n");
   }
   return 0;
}
```

---

**1. Arrays**

**Question 1: Find the Maximum Element in an Array**

**Problem:**
Write a C function to find the largest element in an array of integers.

**Code:**

```
int findMax(int arr[], int n) {
   int max = arr[0];
   for (int i = 1; i < n; i++) {
      if (arr[i] > max)
         max = arr[i];
   }
   return max;
}
```

**Question 2: Reverse an Array**

**Problem:**
Write a C function to reverse an array of integers.
**Example:**
Input: {1, 2, 3, 4, 5}
Output: {5, 4, 3, 2, 1}

**Code:**

```
void reverseArray(int arr[], int n) {
    int start = 0, end = n - 1;
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}
```

---

**2. Linked Lists**

**Question 3: Find the Length of a Linked List**

**Problem:**
Write a C function to find the number of nodes in a linked list.

**Code:**

```
int length(struct Node* head) {
    int count = 0;
    struct Node* current = head;
    while (current != NULL) {
        count++;
        current = current->next;
    }
    return count;
}
```

**Question 4: Reverse a Linked List**

**Problem:**
Write a C function to reverse a singly linked list.

**Code:**

```
struct Node* reverse(struct Node* head) {
    struct Node *prev = NULL, *current = head, *next = NULL;
```

```
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}
```

---

**3. Stacks**

**Question 5: Implement Stack Using Array**

**Problem:**
Write a C program to implement a stack using an array with the following operations: push, pop, and display.

**Code:**

```
#define MAX 100
int stack[MAX], top = -1;

void push(int data) {
    if (top == MAX - 1) {
        printf("Stack overflow\n");
    } else {
        stack[++top] = data;
    }
}

int pop() {
    if (top == -1) {
        printf("Stack underflow\n");
        return -1;
    } else {
        return stack[top--];
```

```
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
```

---

**4. Queues**

**Question 6: Implement Queue Using Array**

**Problem:**
Write a C program to implement a queue using an array with the following operations: enqueue, dequeue, and display.

**Code:**

```
#define MAX 100

int queue[MAX], front = -1, rear = -1;

void enqueue(int data) {
    if (rear == MAX - 1) {
        printf("Queue overflow\n");
    } else {
        if (front == -1) front = 0;
        queue[++rear] = data;
    }
}
```

```
int dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue underflow\n");
        return -1;
    } else {
        return queue[front++];
    }
}

void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty\n");
    } else {
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}
```

---

**5. Sorting**

**Question 7: Implement Bubble Sort**

**Problem:**
Write a C function to sort an array using bubble sort.

**Code:**

```
void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
```

```
            arr[j + 1] = temp;
        }
    }
}
```

**Question 8: Implement Selection Sort**

**Problem:**
Write a C function to sort an array using selection sort.

**Code:**

```
void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex])
                minIndex = j;
        }
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
```

---

**6. Searching**

**Question 9: Implement Binary Search**

**Problem:**
Write a C function to implement binary search on a sorted array.

**Code:**

```
int binarySearch(int arr[], int n, int target) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == target)
```

```
            return mid;
        else if (arr[mid] < target)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;  // Element not found
}
```

---

**7. Recursion**

**Question 10: Find the Factorial of a Number Using Recursion**

**Problem:**
Write a recursive C function to find the factorial of a number.

**Code:**

```
int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```

**Question 11: Fibonacci Series Using Recursion**

**Problem:**
Write a recursive C function to find the nth Fibonacci number.

**Code:**

```
int fibonacci(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
```

---

**8. Arrays**

**Question 12: Find the Second Largest Element in an Array**

**Problem:**
Write a C program to find the second largest element in an array of integers.
**Example:**
Input: {12, 35, 1, 10, 34, 1}
Output: 34

---

**Question 13: Rotate an Array to the Right by k Positions**

**Problem:**
Write a C program to rotate an array of n elements to the right by k positions.
**Example:**
Input: {1, 2, 3, 4, 5}, k = 2
Output: {4, 5, 1, 2, 3}

---

**Question 14: Find the Missing Number in an Array of n-1 Integers**

**Problem:**
Write a C program to find the missing number in an array of size n-1 containing integers from 1 to n.
**Example:**
Input: {1, 2, 4, 5, 6}, n = 6
Output: 3

---

**Question 15: Merge Two Sorted Arrays**

**Problem:**
Write a C program to merge two sorted arrays into one sorted array.
**Example:**
Input: {1, 3, 5}, {2, 4, 6}
Output: {1, 2, 3, 4, 5, 6}

---

**9. Linked Lists**

**Question 16: Detect a Loop in a Linked List**

**Problem:**
Write a C function to detect if a linked list contains a loop. If a loop exists, find the starting node of the loop.

---

**Question 17: Merge Two Sorted Linked Lists**

**Problem:**
Write a C program to merge two sorted linked lists into one sorted linked list.

---

**Question 18: Find the Middle Element of a Linked List**

**Problem:**
Write a C function to find the middle element of a linked list in a single traversal.
**Example:**
Input: 1 -> 2 -> 3 -> 4 -> 5
Output: 3

---

**Question 19: Delete a Node Without Using Head Pointer**

**Problem:**
Write a C function to delete a node from a linked list, given only a pointer to that node.
**Example:**
Input: Linked list: 1 -> 2 -> 3 -> 4, Node to delete: 3
Output: Linked list: 1 -> 2 -> 4

---

**10. Stacks**

**Question 20: Implement a Stack Using a Linked List**

**Problem:**
Write a C program to implement a stack using a singly linked list. Implement the push, pop, and display operations.

---

**Question 21: Check for Balanced Parentheses Using Stack**

**Problem:**
Write a C program to check if an expression has balanced parentheses using a stack.
**Example:**
Input: "((a+b)*(c-d))"
Output: Balanced

---

**Question 22: Reverse a Stack Using Recursion**

**Problem:**
Write a C program to reverse a stack using recursion.

---

**11. Queues**

**Question 23: Implement a Circular Queue Using Array**

**Problem:**
Write a C program to implement a circular queue using an array. Implement enqueue, dequeue, and display operations.

---

**Question 24: Implement a Queue Using Two Stacks**

**Problem:**
Write a C program to implement a queue using two stacks. Implement enqueue and dequeue operations.

---

**12. Trees**

**Question 25: Find the Height of a Binary Tree**

**Problem:**
Write a C function to find the height of a binary tree.
**Example:**
Input:

```
    1
   / \
  2   3
 / \
4   5
```

Output: 3

---

**Question 26: Print Inorder Traversal of a Binary Tree**

**Problem:**
Write a C function to print the inorder traversal of a binary tree (Left-Root-Right).
**Example:**
Input:

```
    1
   / \
  2   3
 / \
4   5
```

Output: 4 2 5 1 3

---

**Question 27: Check if a Binary Tree is a Mirror of Itself**

**Problem:**
Write a C program to check if a binary tree is a mirror of itself (symmetric around its center).

---

**Question 28: Find the Lowest Common Ancestor (LCA) in a Binary Tree**

**Problem:**
Write a C function to find the lowest common ancestor of two nodes in a binary tree.

---

**13. Sorting**

**Question 29: Implement Quick Sort**

**Problem:**
Write a C program to implement quick sort.

---

**Question 30: Implement Merge Sort**

**Problem:**
Write a C program to implement merge sort.

---

**Question 31: Find the kth Smallest Element Using Quick Select**

**Problem:**
Write a C program to find the kth smallest element in an unsorted array using the quick select algorithm.
**Example:**
Input: {7, 10, 4, 3, 20, 15}, k = 3
Output: 7

---

**14. Recursion**

**Question 32: Solve Tower of Hanoi**

**Problem:**
Write a C program to solve the Tower of Hanoi problem for n disks.
**Example:**
Input: n = 3
Output:

Move disk 1 from A to C

Move disk 2 from A to B

Move disk 1 from C to B

Move disk 3 from A to C

Move disk 1 from B to A

Move disk 2 from B to C

Move disk 1 from A to C

---

**Question 33: Generate All Subsets of a Set Using Recursion**

**Problem:**
Write a C program to generate all subsets (power set) of a given set using recursion.
**Example:**
Input: {a, b, c}
Output: { }, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}

---

**15. Miscellaneous**

**Question 34: Find the Majority Element in an Array**

**Problem:**
Write a C program to find the majority element in an array, i.e., the element that appears more than n/2 times.
**Example:**
Input: {3, 3, 4, 2, 4, 4, 2, 4, 4}
Output: 4

---

**Question 35: Find the Intersection of Two Arrays**

**Problem:**
Write a C program to find the intersection of two arrays.
**Example:**
Input: {1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}
Output: {4, 5}

---

**Question 36: Implement a Min-Heap**

**Problem:**
Write a C program to implement a min-heap and its basic operations: insert, delete, and extractMin.