50 Logical Reasoning Questions in C Programming

Section 1: Output Prediction

Q1. int a = 5;
printf("%d", a++ + ++a);

ans: 12

Q2. int a = 10;
if (a = 5)
printf("True");
else
printf("False");

ans: true

Q3. int x = 10;
printf("%d", x == 10 ? 100 : 200);

ans:100

Q4. int a = 5, b = 5;
if (a++ == ++b)
printf("Equal");
else
printf("Not Equal");

ans:Not Equal

Q5. int x = 3;
x = x << 1;
printf("%d", x);

ans:6

Q6. int a = 10;
int b = a++ + ++a;
printf("%d", b);

ans:22

Q7. int arr[] = {1, 2, 3, 4};
printf("%d", *(arr + 2));

ans:3

Q8. int x = 0;
while (x < 3)
printf("%d", x++);

ans:0,1,2

Q9. int x = 2;
printf("%d", x++ * x++);

ans: 6

Q10. int a = 5;
int b = 10;
printf("%d", a > b ? a : b);

ans: 10

Q11. char str[] = "Hello";
printf("%c", *str);

ans: H

Q12. int a = 1;
int b = 1;
if (a-- && b++)

ans: yes

Q13. int a = 5;
printf("%d", sizeof(a++));

ans: 4

Q14. int x = 1;
do {
printf("%d", x);
} while (x-- > 1);

ans: 1

Q15. int a = 5;
int *p = &a;
*p = *p + 1;
printf("%d", a);

ans: 6

Section 2: Code Logic Understanding

Q1. int x = 0;
if (x = 10)
printf("Yes");

ans: The condition (x = 10) is an assignment, not a comparison. It assigns 10 to x, and the result of the assignment (10) is then evaluated by the if statement. Since 10 is non-zero, it's treated as true, and "Yes" is printed.

Q2. const int x = 10;
x = 20;

Ans: The compiler will flag this as an error because you are trying to modify a constant value.

Q3. int *ptr;
*ptr = 5;

ans: that ptr is uninitialized, meaning it points to an unknown or invalid memory location, and attempting to dereference it will cause a runtime error.

Q4. Can a function return an array in C?

Ans :a function cannot return an array in C.

Q5. int a = 5;
{
int a = 10;
}

ans : we could print 5 only at the outside of the curly braces, if we want to print 10 we should declare print statement inside the curly braces.

Q6. What happens if you access an array out of bounds?

Ans: it returns the segmentation error .

Q7. int i;
for (i = 0; i < 10; i++);
printf("%d", i);

ans: the semicolon after the for loop is define the loop is don’t have the body ,the loop increce only the increment .then print i.

Q8. int a = 5;
printf("%d", ++a++);

ans: the variable couldn’t perform the multiple operation at a time.

Q9. int a = 1, b = 2;
int c = a+++b;

ans: the c variable takes the a+++b as (a++)+b;

Q10. Which statement is true about `static` keyword?

Ans: variable declared `static` inside a function retains its value between function calls. It's initialized only once.
\\

Q11. int i = 0;
while (i++ < 5)
continue;
printf("%d", i);

ans: After the loop finishes, the value of i is 5. This value is then printed.then print the i

Q12. What is `NULL`?

ans :NULL is primarily used to indicate that a pointer variable does not point to any valid memory location.

Q13. Which one is correct to declare a function returning pointer to int?

Ans: int* functionName(parameters);

Q14. When does a segmentation fault occur in C?

ans: when a program attempts to access a memory location that it is not allowed to access, or attempts to access memory in a way that is not allowed

Q15. Can a `void` function return a value?

Ans: void function cannot return a value in C.

Section 3: Pointers and Memory

Q1. int a = 5;
int *p = &a;
printf("%d", *p);

ans: p is an dereference variable it hold the address of a,while print the value at p points the value at address of a.it prints 5

Q2. int *ptr = NULL;
printf("%d", *ptr);

ans: a segmentation fault, because it attempts to dereference a NULL pointer, accessing memory it's not allowed to.

Q3. What is the difference between malloc() and calloc()?

Ans :malloc() allocates memory without initializing it (contains garbage values), while calloc() allocates memory and initializes all bytes to zero.

Q4. What does `free()` do?

Ans :free() deallocates previously allocated dynamic memory

Q5. int *p;
*p = 10;

ans :Segmentation fault ,cause of invalid memory location.

Q6. Can a pointer point to a function?

Ans :yes, a pointer can point to a function. This is known as a function pointer, allowing functions to be passed as arguments or stored in variables.

Q7. What is a dangling pointer?

Ans : dangling pointer is a pointer that points to a memory location that has been deallocated (freed) or no longer exists. Dereferencing it leads to undefined behavior.

Q8. How do you allocate a 2D array dynamically?

Ans : 2D array is dynamically allocated by creating an array of pointers, where each pointer then points to a dynamically allocated row. This typically involves nested malloc calls.

Q9. What happens if you free() memory twice?

Ans :freeing memory twice (double-free) leads to undefined behavior, which can corrupt the heap, cause crashes (segmentation faults), or create security vulnerabilities.

Q10. int x = 10;
int *p = &x;
int **q = &p;
printf("%d", **q);

ans :it prints 10,becoz the dereference of value at q hold the dereference of p's address ,that p hold the address of x .

Section 4: Conceptual & Logical
Q1. Why is `main()` special in C?

Ans:It's where the program begins running and the primary interface for system interaction.

Q2. What is the output type of `sizeof()`?

Ans : unsigned integer

Q3. printf("%d", sizeof('A'));

ans: 4

Q4. What are header files used for?
Ans : header files are used for call the predefined functions and keywords

Q5. Difference between `==` and `=` in C?

Ans :'==' it's an comparision operator,'=' it's an assignment operator.

Q6. What happens when `break` is used in a loop?

Ans : When break is encountered inside a loop it immediately terminates the innermost loop containing it.

Q7. What is a function pointer?

Ans :A function pointer is a variable that stores the memory address of a function.

Q8. Difference between `int func()` and `int func(void)`?

Ans :
int func(): In C, this declares a function func that returns an int but takes an unspecified number of arguments
int func(void): This explicitly declares a function func that returns an int and takes no arguments.

Q9. What does `volatile` mean?

Ans :value can change at any time

Q10. int a = 1;
int b = 0;
printf("%d", a && b || !a);

ans : 0