Qno.1)

1. Program 1:

Output: 20

Explanation: The function fun takes a pointer as an argument and assigns the address of a local variable q to the pointer. This doesn't affect the value of the pointer p in main, because the argument is passed by value. Therefore, the printf statement in main prints the original value pointed by p, which is 20.

1. Program 2:

Output: 10

Explanation: The function fun takes a pointer to a pointer as an argument, and assigns the address of the static variable q to the pointer. This effectively changes the address that p in main is pointing to. As a result, the printf statement in main prints the value of q, which is 10.

1. Program 3:

Output: 2 5

Explanation: The pointer ptr is assigned to the address of the array a incremented by 1, which points to one element after the array. \*(a+1) refers to the second element of the array, which is 2. \*(ptr-1) refers to the last element of the array, which is 5.

1. Program 4:

Output: TEST G QUIZ MCQ

Explanation: The program manipulates pointers to pointers to pointers to character arrays, and the printf statements access different elements and parts of those elements, resulting in the given output.

1. Program 5:

Output: GeeksQuiz

Explanation: The fun function increments the pointer str\_ref, but the passed argument is a pointer to a pointer, which doesn't affect the original pointer str. Therefore, the puts statement in main prints the original string "GeeksQuiz".

1. Program 6:

Output: ef

Explanation: The function f takes a pointer to a pointer as an argument, manipulates the pointer, and then prints the resulting string. The output is "ef", which is the third element in the argv array.

1. Program 7:

Output: 2 3 5 6

Explanation: The program creates a pointer to an array of 3 integers, initializes it with a 2D array a, and then increments the pointer to print values from the two arrays in a.

1. Program 8:

Output: 0 1 1 2 3

Explanation: This program allocates memory for 5 integers, assigns values to them, and then manipulates the pointer in different ways using pre-increment and post-increment operators in the printf statements.

1. Program 9:

Output: 20 10

Explanation: The fun function takes an array as an argument, increments the pointer, and then prints the second element of the array (20). The printf statement in main prints the first element of the array (10).

Qno.2)

1. Program 1:

Output: Value of a is 10

Explanation: This program demonstrates a simple use of function pointers. A function pointer fun\_ptr is declared with the same signature as the fun function. The address of the fun function is assigned to fun\_ptr. Then, the fun function is called through the function pointer fun\_ptr with the argument 10.

1. Program 2:

Output: Depends on the user's input

Explanation: This program demonstrates an array of function pointers. The array fun\_ptr\_arr contains pointers to three functions: add, subtract, and multiply. Based on the user's input (0, 1, or 2), the corresponding function is called with the arguments a and b (15 and 10).

1. Program 3:

Output:

Fun1

Fun2

Explanation: This program demonstrates passing a function pointer to another function. The wrapper function takes a function pointer as an argument and calls the function through the pointer. In main, the wrapper function is called with two different functions, fun1 and fun2, resulting in the output "Fun1" and "Fun2".

1. Program 4:

Output: 5 10 12 15 80 90

Explanation: This program demonstrates the use of function pointers with the qsort function from the C Standard Library. The compare function is defined to sort integers in ascending order. The qsort function takes the array, its size, element size, and a pointer to the compare function as arguments. After sorting the array, the program prints the sorted array.

1. Program 5:

Output: Returned index is 2

Explanation: This program demonstrates the use of function pointers with a custom search function. The compare function compares two integers for equality. The search function takes an array, its size, element size, a value to search, and a pointer to the compare function as arguments. The program searches for the value x (7) in the array arr and prints the returned index (2).

Qno.3)

1. Program 1:

Output:

\*ptr1 = 10

\*ptr2 = 10

Explanation: This program demonstrates a function that returns a pointer. The function returnPointer takes a pointer as an argument and returns the same pointer. In the main function, two pointers ptr1 and ptr2 are created, where ptr1 points to i and ptr2 receives the pointer returned by returnPointer. Both pointers point to the same memory location, so the output shows the same value (10) for both pointers.

1. Program 2:

Output: Undefined behavior

Explanation: This program demonstrates a function that returns a pointer to a local variable. The function returnFunc declares a local variable i and returns its address. However, this is not safe because the local variable's lifetime ends when the function returns. Accessing the local variable through the returned pointer in the main function will lead to undefined behavior.

1. Program 3:

Output:

\*ptr = 10

\*ptr = 10

\*ptr = 10

Explanation: This program demonstrates a function that returns a pointer to a static variable. The function returnFunc declares a static variable i and returns its address. Since the static variable's lifetime extends to the entire program's execution, it is safe to access it through the returned pointer in the main function. The output shows the same value (10) for each printf statement, as the pointer points to the same static variable.