1. What is the purpose of the return statement in a function in C?
   a) To terminate the function and return control to the calling function
   b) To assign a value to a variable
   c) To print a message to the console
   d) To declare a variable

Answer: (a) To terminate the function and return control to the calling function

2. What is a function prototype in C?
   a) A function definition that includes the function name, parameters, and body
   b) A statement that declares the name, return type, and parameters of a function
   c) A block of code that is executed when a function is called
   d) A reserved keyword that specifies the type of a variable

Answer: (b) A statement that declares the name, return type, and parameters of a function

3. What is the difference between a function declaration and a function definition in C?
   a) A declaration specifies the function name, return type, and parameters, while a definition includes the function body.
   b) A declaration includes the function body, while a definition specifies the function name, return type, and parameters.
   c) There is no difference between a declaration and a definition in C.
   d) None of the above.

Answer: (a)

4. A function prototype is used for
   a) Declaring the function logic
   b) Calling the function from the main body
   c) Telling the compiler, the kind of arguments used in the function
   d) Telling the user for proper use of syntax while calling the function

Solution: (c) A function prototype tells the compiler what kind of arguments a function is expecting to receive and what type of return value a function will give back. This approach helps the compiler ensure that calls to a function are made correctly and that no erroneous type conversions occur.

5. What is the scope of a variable declared inside a function in C?
   a) The variable can be accessed by any function in the program
   b) The variable can only be accessed within the function where it is declared
   c) The variable can be accessed by any function in the same file
   d) The variable can be accessed by any function in the same module

Answer: (b) The variable can only be accessed within the function where it is declared

6. What is the output of the following C program?

```
#include <stdio.h>
void foo(), f();
int main()
{
   f();
return 0;
}
void foo()
{
    printf("2 ");
}
void f()
{
    printf("1 ");
    foo();
}
```

   a) Compiler error as foo() is not declared in main
   b) 1 2
   c) 2 1
   d) Compile-time error due to declaration of functions inside main

Answer: (b) The function f() is called from the main. After printing '1', foo() will be called, and '2' will be printed. This is an example of a nested function.

7. What is the output of the following code snippet?

```
#include <stdio.h>
void swap(int a, int b) {
  int temp = a;
  a = b;
  b = temp;
}

int main() {
  int x = 1, y = 2;
  swap(x, y);
  printf("x = %d, y = %d\n", x, y);
  return 0;
}
```

   a) x = 1, y = 2
   b) x = 2, y = 1
   c) Compilation error
   d) Runtime error

Answer: (a) x = 1, y =2

This is an example of pass by value in C. In ‘swap’ function the values gets swapped locally, i.e. inside the swap function, but the actual value remains the same in ‘main’ i.e. x = 1, y = 2. Therefore, when printed, the same value gets produced. To make permanent changes of the variables, you need to use pass by argument operation.

8. What is the error in the following program

```
#include<stdio.h>
int f(int a)
{
  a > 20? return(10): return(20);
}
int main()
{
    int b;
    b = f(20);
    printf("%d\n", b);
return 0;
}
```

   a) Error: Return statement cannot be used with conditional operators
   b) Error: Prototype declaration
   c) Error: Two return statements cannot be used in any function
   d) No error

Answer: (a) We cannot use the return statement in a ternary operator. The ternary operator requires expressions but not code.

```
#include<stdio.h>
int f(int a)
{
  int x = a > 20? 10: 20;
  return(x);
}
int main()
{
    int b;
    b = f(20);
    printf("%d\n", b);
 return 0;
}
```

9. What is the output of the following?

```
#include <stdio.h>
int fun(int n) {
  if (n <= 0) {
    return 0;
  }
  return n + fun(n - 1);
}

int main() {
  int result = fun(10);
  printf("%d\n", result);
  return 0;
}
```

Answer: 55

This is an example of recursion in C. This is how the steps get executed.

fun(10) → 10 + fun(9) → 10 + 9 + fun(8) → 10 + 9 + 8 + fun(7) →……

this way the recursion tree gets evaluated. Finally, the sum of 10 to 0 is produced as the result, which is 55.

10. Consider the function

```
int fun(int x)
{
   int y;
   if(x > 100)
       y = x-10;
   else
       y = fun(fun(x + 11));
   return y;
}
```

For the input x = 95, the function will return

a) 89
b) 90
c) 91
d) 92

Solution: (c)

f(95)–>f(f(106)) = f(96)–>f(f(107)) = f(97)–>f(f(108)) = f(98)–>f(f(109)) = f(99)–>f(f(110)) = f(100)–>f(f(111))=f(101)=91

So the correct option is (c)