Chapter 7 – Functions

✅ Short Questions with Easy Answers

Q1. Define function? Why is it used in a program?

👉 A function is a **named block of code** that performs a task.

- It runs only when **called**.
- Functions make programs easier to manage by dividing them into parts.

Q2. How does function make programming easier?

👉 A big program is divided into **small functions**, so:

- Easier to write and understand.
- Easier to debug (fix errors).
- Focus on one problem at a time.

Q3. List some benefits of using functions.

1. Easier to code
2. Easier to modify
3. Easier to maintain & debug
4. Reusability (use again in other programs)

Q4. List different types of function in C.

1. **User-defined functions** → written by programmer.
2. **Built-in functions (library functions)** → already available in C (e.g., printf(), sqrt()).

Q5. Describe built-in functions.

👉 Ready-made functions provided by C.

- Stored in **header files** (e.g., math.h, stdio.h).
- Example:

```
printf("Hello");   // stdio.h
sqrt(16);          // math.h
```

Q6. Define function body.

👉 The set of statements inside { } that perform the task.

Example:

```
void greet() {
  printf("Hello World");
}
```

Q7. What is function declaration or prototype?

👉 A function **model** that tells the compiler function's name, return type, and parameters.

Example:

```
int add(int a, int b);
```

Q8. What is function definition?

👉 Actual code of the function (header + body).

Example:

```
int add(int a, int b) {
  return a + b;
}
```

Q9. Differentiate between function definition and declaration.

Function Declaration (Prototype)	**Function Definition**
Just tells compiler function name, return type, parameters	Actual block of code
Ends with ;	Has { } with statements
Example: int add(int, int);	Example: int add(int a, int b) { return a+b; }

Q10. What is a function call?

👉 Statement that **activates a function**.

Example:

```
int result = add(5, 3);   // function call
```

✅ **Long Questions with Easy Answers & Examples**

Q1. Briefly explain the benefits of using functions.

- Breaks large programs into **small parts**.
- **Reusability**: same function used many times.
- **Easy debugging**: find and fix errors quickly.
- **Teamwork**: different programmers can work on different functions.

Q2. Explain importance of functions.

- Functions **reduce repetition**.
- Save time in programming.
- Improve program **readability** and **structure**.
- Useful in real-life projects like banking software, billing systems, etc.

Q3. Explain function body and function header.

- **Function header** → name, return type, parameters.

```
int add(int x, int y);   // header (prototype)
```

- **Function body** → code inside { }.

```
int add(int x, int y) {   // definition
  return x + y;
}
```

Q4. Program: Input two numbers in main and pass them to a function. The function displays first number raised to the power of second number.

```
#include <stdio.h>
#include <math.h>   // for pow()

// Function Declaration
void power(int base, int exp);

int main() {
  int a, b;
  printf("Enter base: ");
  scanf("%d", &a);
  printf("Enter exponent: ");
  scanf("%d", &b);

  power(a, b);   // Function call
  return 0;
}

// Function Definition
void power(int base, int exp) {
  int result = pow(base, exp);
  printf("%d ^ %d = %d", base, exp, result);
}
```

Input: 2 3
Output: 2 ^ 3 = 8

Q5. Explain local variable.

👉 A variable declared **inside a function** is called a **local variable**.

- It is **only accessible within that function**.
- Memory is allocated when function starts, and released when function ends.

Example:

```
void test() {
  int x = 5;    // local variable
  printf("%d", x);
}
```

Here, x is **local** to test().

🔷 **Extra External Questions with Answers**

Q1. What is the difference between local and global variable?

- **Local variable** → declared inside function, used only in that function.
- **Global variable** → declared outside all functions, can be used by all functions.

Example:

```
int g = 10;   // global variable
void test() {
  int l = 5; // local variable
  printf("%d %d", g, l);
}
```

Q2. Write a function to find factorial of a number.

```
int factorial(int n) {
  int fact = 1;
  for (int i = 1; i <= n; i++) {
    fact *= i;
  }
  return fact;
}
```

Input: 5 → **Output:** 120

Q3. What is recursion? Give example.

👉 A function that **calls itself** is called recursive function.

Example:

```
int factorial(int n) {
  if (n == 0) return 1;

  return n * factorial(n-1);

}
```