CHAPTER 13 (Functions in C)

Q 1. What is modular programming?

Ans. A programming technique in which a program consists of many independent parts is called modular programming. These parts are called modules. These parts are also called function. Each module can perform different tasks. Different modules are combined to make a complete program.

Q 2. What is a function?

Ans. In structured programming the program consists of more than more one part. Each part of program is called a module or function. Function can be defined as " A named piece of code developed to perform a specific task is called function".

Q 3. Why functions are used? / Benefits/Advantages of function.

Ans. Function is a piece of code designed to perform a specific task. These advantages are described below:

- Easy programming
- Easy modification
- Easy debugging
- Reuse-ability
- Eliminates duplicate code
- Less programming time

Q 4. What are built-in functions?

Ans. The function that are provided as a part, C language are called built-in functions. These functions are also called library function. A large number of built-in function are provided by C language. These functions are stored in different header files. Examples are printf() scanf().

Q 5. What are user defined functions?

Ans. The functions that are written by the programmer to perform specific task are called user defined functions. These functions are written according to the requirement of the program.

Q 6. What Is function prototypes? / Function declaration?

Ans. Function declaration is also called function prototype. It is a statement that provides basic information to compiler about the structure of the function. Function declaration statement ends with semicolon. Function can be declared before the main() function or inside the main() function.

Syntax:

Return_type function_name (paramters);

Example:

Int add(int a, int b);

Q 7. What is function definition also write its syntax?

Ans. Every function perform some specific task. The task is performed when the set of instructions execute. Writing set of statements of a function is called function definition. Function definition is always done outside main() function.

Syntax:

Return_type function_name (paramters);

Example:
Int add(int a, int b);

Q 8. What Is function header?

Ans. The first line of the function definition is called function header. It is similar to function prototype. The only difference is that it is not terminated with semicolon. Its general **syntax** is as follow:

• Return_Type Function_Name(parameters)

Q 9. What is function calling?

Ans. The statement that is written to use a function is called function call. A function is called by using its name. The required parameters are maintained after the name in braces at the end of the function call statement. Semicolon is used at the end of statement in which function is called.
Example: add(3,5);

Q 10. What is return statement?

Ans. Keyword "return" is used to return a value from the body of called function to calling function. The statement in which "return" keyword is used is called return statement. A function can return a single value.
Syntax
return expression;

Q 11. What is return type?

Ans. The return type in function declaration indicates the type of value returned by a function e.g. int is used as return type if the function returns integer value. If the function returns no value, the keyword void is used as return type

Q 12. What are parameters?

Ans. Parameters are the values that are provided to a function when the function is called. Parameters are given in the parentheses. If there are many parameters, these are separated by commas. If there is no parameter, empty parentheses are used.

- Parameters in function call are called actual parameters/arguments
- Parameters in function declaration are called formal parameters

Q 13. What Is a local variable?

Ans. The variables declared inside curly braces { } (in main function, user defined function or conditional statement etc) are called local variables. Local variable also called automatic variables.
Syntax:
auto data-type variables-name;
The use of auto is optional.

Q 14. What Is global variable?

Ans. The variables that are declared outside the main() function or any other function are called global variable. Global variables can be used by all functions in the program. All functions can share their value. If value As global variable is changes in a function, that changes value is also available in other functions.

Q 15. What is meant by life time of a variable? / life time of local and global variable.

Ans. The time period for which a variable exists in the memory is called life time of a variable.

Lifetime of a **local variable** is limited, when control enters into the function and variable declaration statement is executed, they are created in memory. When the control exits from the function these variables are destroyed and their Life ends. When program starts execution, **global variables** are created in memory. They remain in memory, till the termination of the program.

Q 16. What Is meant by scope of a variable? Scope of local and global variable.

Ans. The area where a variable can be accessed is known as scope of variable. **Local variables** have a limited scope they can only be used in the function in which they are declared. **Global variables** can be accessed in all modules of program. They are accessible in main() function as well as all other user defined functions.

Q 17. What do you know about a function without argument?

The simplest type of function is a function that accepts no argument and returns no value. The return type of such function is void. The parameter list may be empty or the keyword void can be written in the parenthesis. **Example**

```
Void hello()
{
Printf(“Hello world”);
}
```

Note: Prepare Programs from this chapter from book.

OBJECTIVES (MCQ'S) OF CHAPTER-13
IN ALL PUNJAB BOARDS 2011-2021

1. **Functions that are the part of language are called:**
(A) intrinsic (B) built in function (C)language defined (D)all these

2. **printf ( ) is a:** **(2 Times)**
(A) built in function (B) user defined function (C) local function (D) keyword

3. **Memory is allocated to a local variable at the time of its** **(2 Times)**
(A) declaration (B) destruction (C) definition (D) first reference

4. **Local variables are called:** **(2 Times)**
(A) Normal (B)Automatic (C) global (D) none

5. **Global Variable are created in** **(6 Times)**
(A) ROM (B) cache (C) RAM (D) hard disk

6. The first line of function definition is known as :
(A) Function body (B)Function call (C) Function arguments (D) Function header

7. **Multiple arguments passed to a function are separated by:** **(3 Times)**
(A) period (B) colon (C) comma (D) semicolon

8. **Function prototype for built in function are specified in:**
(A) source files (B) header files (C) object files (D)image files

9. **Which of the following is type of function available in C language:**
(A) User-defined (B) Built-in (C)Subprogram (D)Both a and b

10. **Another name for built-in function is:**
(A) User-defined function (B) Library function
(C) Arithmetic function (D) Both a and b

11. **A type of function that is available as part of language is known as:**
(A) User-defined function (B) Library
(C) Sub-program (D) Both a and b

12. **The statement that activates a function is known as:**
(a) Function design (b) Function definition (c) Function declaration (d) Function call

2017

13. **gets ( ) function takes_____parameters.** **(13 times)**
(a) 1 (b) 2 (c) 3 (d) 4

14. **The first line of user defined function definition is:** **(13 times)**
(a) function argument (b) function prototype (c) function header (d) function calling

15. **Function declaration is also known as function.....** **(3 times)**
(a) Definition (b) Header (c) Prototype (d) Parameters

16. **Which statement is used by function to return a value?** **(13 times)**
(a) give (b) send (c) return (d) call

17. **A type of function written by the programmer is known as:**
(a) User-defined (b) Subprograms (c) Subroutines (d) Built-in-function

18. **A value that can be sent to a function is known as:-**
(a) Return value (b) Automatic variable (c) Indicator (d) Argument

2018

19. **What is the variable that is used by function to receive an argument:**
(a) expression (b) parameter (c) constant (d) function

20. **Formal arguments are also called:**
(a) Actual arguments (b) Dummy arguments (c) Original arguments (d) Referenced arguments

21. **A function does not return any value has return type:**
(a) nothing (b) float (c) void (d) int

2019

22. The scope of variable refers to its:
(a) Length (b) Name (c) Accessibility (d) Data type

23. The process of sending an argument to a function is called:
(a) Sending (b) Filtering (c) Delivering (d) Passing

24. The parameters in function declaration:
(a) actual parameters (b) formal parameters
(c) returned parameters (d) call parameters

25. The statement that activates a function is known as:
(a) Function Output (b) Function Definition
(c) Function Prototype (d) Function Call

ANSWERS

1	2	3	4	5	6	7	8	9	10	11	12	13
B	A	A	B	C	D	C	B	D	B	B	D	A
14	15	16	17	18	19	20	21	22	23	24	25	
C	C	C	A	D	B	B	C	C	D	B	D	

Chapter 13 Ex Programs Q#8 to 15

```
// Question 8: Write a function GCD to compute the greatest common divisor.
#include <stdio.h>
#include <math.h>
int GCD(int a, int b) {
   if (b == 0) return a;
   return GCD(b, a % b);
}

int main() {
   int num1, num2;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);

   int gcd = GCD(num1, num2);
   printf("Greatest Common Divisor: %d\n");
}

// Question 9: Write a program to compute the factorial of a number.
int Factorial(int n) {
   if (n <= 1) return 1;
   return n * Factorial(n - 1);
}

int main() {
   int n;
   printf("Enter a number: ");
```

```
    scanf("%d", &n);

    int factorial = Factorial(n);
    printf("Factorial of %d is %d\n", n, factorial);
}

// Question 10: A program for basic arithmetic operations based on user input.
int Add(int a, int b) {
    return a + b;
}

int Subtract(int a, int b) {
    return a - b;
}

int Multiply(int a, int b) {
    return a * b;
}

int Divide(int a, int b) {
    if (b == 0) {
        printf("Error: Division by zero\n");
        return -1;
    }
    return a / b;
}

int main() {
```

```
int choice, num1, num2, result;
printf("Enter two numbers: ");
scanf("%d %d", &num1, &num2);

printf("Enter your choice:\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Exit\n");

scanf("%d", &choice);
switch (choice) {
   case 1:
      result = Add(num1, num2);
      break;
   case 2:
      result = Subtract(num1, num2);
      break;
   case 3:
      result = Multiply(num1, num2);
      break;
   case 4:
      result = Divide(num1, num2);
      break;
   case 5:
      return 0;
   default:
      printf("Invalid choice\n");
      return -1;
}

printf("Result: %d\n", result);
```

```
}
```

// Question 5: Write a function Is_Prime that has an input parameter and returns 1 if the number is prime, otherwise returns 0.

```
#include <stdio.h>

int Is_Prime(int num) {
    if (num <= 1) return 0; // 0 and 1 are not prime
    for (int i = 2; i <= sqrt(num); i++) {
        if (num % i == 0) {
            return 0; // Not prime
        }
    }
    return 1; // Prime
}

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (Is_Prime(num)) {
        printf("%d is prime.\n");
    } else {
        printf("%d is not prime.\n");
    }
}
```

```
// Question 4: Write a function named Draw_Asterisks that prints asterisks in a pattern.
#include <stdio.h>
void Draw_Asterisks(int n) {
    for (int i = n; i >= 1; i--) {
        for (int j = 1; j <= i; j++) {
            printf("*");
        }
        printf("\n");
    }
}

int main() {
    Draw_Asterisks(9);
}

// Question 3: Write a program to reverse a number using a function.
#include <stdio.h>
int Reverse(int num) {
    int reversed = 0;
    while (num > 0) {
        reversed = reversed * 10 + num % 10;
        num /= 10;
    }
    return reversed;
}

int main() {
    int num;
```

```
    printf("Enter a number: ");
    scanf("%d", &num);

    int reversed = Reverse(num);
    printf("Reversed number: %d\n", reversed);
}

// Question 1: Write a program that constructs a rectangle using horizontal and vertical lines.
#include <stdio.h>

void Draw_Horizontal(int width) {
    for (int i = 0; i < width; i++) {
        printf("*");
    }
    printf("\n");
}

void Draw_Vertical(int height) {
    for (int i = 0; i < height; i++) {
        printf("*\n");
    }
}

int main() {
    int width, height;
    printf("Enter the width of the rectangle: ");
    scanf("%d", &width);
    printf("Enter the height of the rectangle: ");
```

```
    scanf("%d", &height);

    printf("Rectangle:\n");
    Draw_Horizontal(width);
    Draw_Vertical(height - 2);
    Draw_Horizontal(width);
}
```