PART A

Q1 Type Casting

Type casting is the conversion of one data type to another.
It allows programming languages to manage different types of data effectively.

Q2 Relational Operator

A relational operator compares two values and returns a boolean result (true or false).
Common relational operators include <, >, <=, >=, ==, and !=.

Q3 Syntax of for and while loop

For loop: for (initialization; condition; increment/decrement) { // code block }
While loop: while (condition) { // code block }

Q4 Difference between while and do-while

A while loop checks the condition before executing the loop body, while a do-while loop checks it after, ensuring the loop body runs at least once.

Q5 Pseudocode

Pseudocode is a high-level description of an algorithm written in plain language.
It represents the logic without the syntax of specific programming languages.

Q6 Flow Chart Annotations

Flowcharts use specific symbols to represent processes (rectangles), decisions (diamonds), inputs/outputs (parallelograms), and flow lines (arrows).

Q7 Declaring and Initializing 1-D and 2-D array

1-D Array: int arr[5] = {1, 2, 3, 4, 5};
2-D Array: int arr[2][2] = {{1, 2}, {3, 4}};

Q8 Recursion

Recursion is a programming technique in which a function calls itself to solve a problem by breaking it down into smaller subproblems.

Q9 #include and Define Directives

#include is used to include header files in C.
#define is used for defining constants or macros.

Q10 Syntax of Function

return_type function_name(parameter_list) { // code block }

Q11 Pointer Array

A pointer array is an array where each element is a pointer to a variable or a memory location.

Q12 Array of Strings

An array of strings is an array where each element is a string (character array).

Q13 Discriminate puts() and gets()

puts() prints a string with a newline, while gets() reads a line of text until a newline.

Q14 Comparing Two Strings

C provides the strcmp() function for comparing two strings lexicographically.

Q15 Data Types in C

Common data types include int, float, char, and double.

PART B
Q2. Structure of C Program & Swapping without Temporary Variable

Structure of C Program:

```
#include <stdio.h>

void main() {
    // Variable Declaration
    // Logic
    // Output
}
```

#include <stdio.h> → Used to include standard input/output library functions.

main() → Starting point of a C program.

Variable declaration → Declares data types and names.

Logic → The program logic is implemented here.

Output → Results are printed using printf().

Program to swap two numbers without using temp:

```
#include <stdio.h>
int main() {
    int a = 10, b = 20;

    printf("Before swapping: a = %d, b = %d\n", a, b);

    a = a + b;
    b = a - b;
    a = a - b;

    printf("After swapping: a = %d, b = %d\n", a, b);
    return 0;
}
```

Q3. Define Algorithm and Pseudocode with Factorial Example

Algorithm:
A step-by-step procedure to solve a problem in a finite number of steps.

Pseudocode:
A human-readable representation of an algorithm using simple English-like statements.

Algorithm to find factorial:

```
    Start
    Read number n
    Set fact = 1
    Repeat i = 1 to n
       → fact = fact * i
    Print fact
    Stop
```

Pseudocode:

```
START
Input n
Set fact = 1
FOR i = 1 TO n
    fact = fact * i
END FOR
Print fact
STOP
```

Q4. Branching Statements in C & Prime Number Program

Types of Branching Statements:

- if
- if-else
- nested if
- switch

Example:

```c
if (a > b)
    printf("a is greater");
else
    printf("b is greater");
```

Program to generate prime numbers between 1 and n:

```c
#include <stdio.h>
int main() {
    int i, j, n, isPrime;
    printf("Enter n: ");
    scanf("%d", &n);

    for (i = 2; i <= n; i++) {
        isPrime = 1;
        for (j = 2; j <= i/2; j++) {
            if (i % j == 0) {
                isPrime = 0;
                break;
            }
        }
        if (isPrime)
            printf("%d ", i);
    }
    return 0;
}
```

Q5. Define Array and Sorting Program

Array:
An array is a collection of elements of the same data type stored at contiguous memory locations.

Initialization of 1D array:

```c
int arr[5] = {10, 20, 30, 40, 50};
```

Sorting in Ascending Order (Bubble Sort):

```c
#include <stdio.h>
int main() {
    int arr[5] = {30, 10, 50, 20, 40};
    int i, j, temp;

    for (i = 0; i < 5; i++) {
        for (j = i + 1; j < 5; j++) {
            if (arr[i] > arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    printf("Sorted Array: ");
    for (i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

Q6. Features of Pointers & Print Address

Features of Pointers:

- Store address of another variable
- Used for dynamic memory allocation
- Efficient in arrays and strings
- Used in call by reference
- Increases performance and flexibility

Program to print address of a variable:

```
#include <stdio.h>
int main() {
    int a = 10;
    int *p = &a;

    printf("Value of a: %d\n", a);
    printf("Address of a: %p\n", p);

    return 0;
}
```

Q7. Structure, Syntax, and Comparison

Structure Definition:
A structure is a user-defined data type in C that allows grouping variables of different types.

Syntax:

```
struct Student {
    int roll;
    char name[20];
};

struct Student s1;
s1.roll = 1;
```

Copying Structures:

```
struct Student s2 = s1;
```

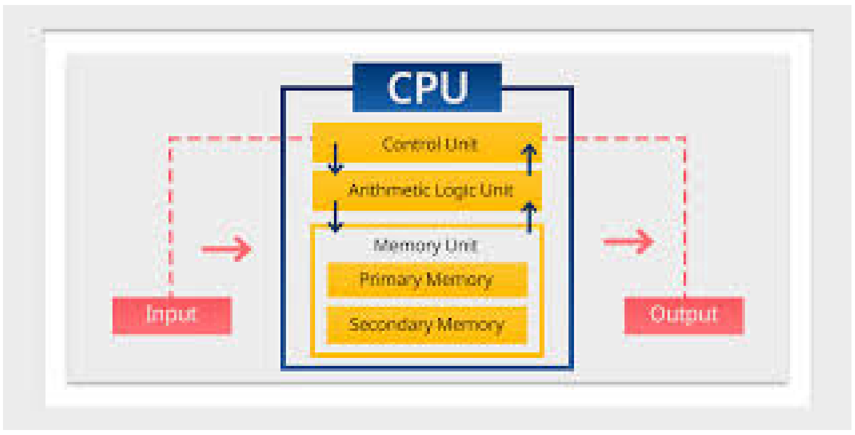
Comparing Structures (Manually):

```
if (s1.roll == s2.roll && strcmp(s1.name, s2.name) == 0)
    printf("Equal");
```

PART C

Q8. Illustrate computer with a neat and clean diagram. Explain their functional units in detail.

Diagram of Computer Functional Units:



Explanation of Units:

1. Input Unit:

    Accepts data and instructions from user.

    Converts them into a form the computer can understand.

    Example: Keyboard, Mouse, Scanner.

2. Central Processing Unit (CPU):

    Brain of the computer.

    Consists of two major parts:
    a) Arithmetic Logic Unit (ALU):

        Performs arithmetic operations (+, −, *, /).

        Performs logical operations (AND, OR, NOT).
    b) Control Unit (CU):

        Controls all operations of the computer.

        Directs flow of data between units.

3. Memory Unit:

    Stores data and instructions before, during, and after processing.

    Divided into:

        Primary Memory (RAM, ROM)

        Secondary Memory (Hard Disk, SSD)

4. Output Unit:

Displays result to the user after processing.

Converts binary to human-readable form.

Example: Monitor, Printer, Speaker.

Summary:

All functional units of a computer work together to accept input, process it, store it, and produce output. The CPU is central to this operation.

Q9. What is the difference between algorithm, pseudocode and flow chart? Draw a flowchart to check if a number is prime or not.

1. Algorithm:

Step-by-step procedure.

Written in simple English.

Example:

Step 1: Start

Step 2: Read n

Step 3: Check if divisible by any number between 2 and n/2

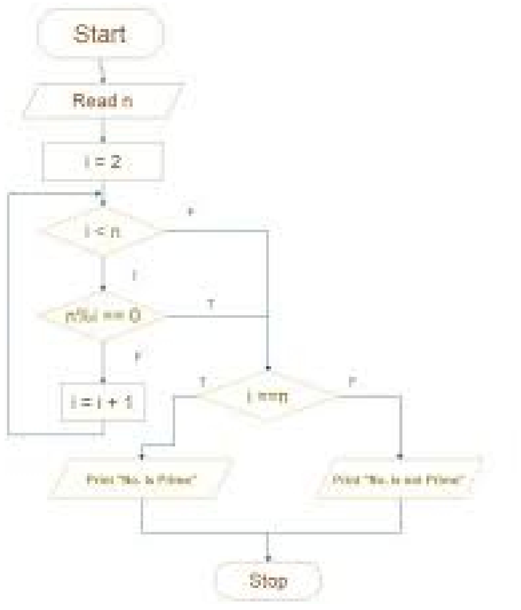
2. Pseudocode:

Similar to algorithm but more programming-like.

Not strict syntax.

```
Input n
flag = 0
FOR i = 2 to n/2
  IF n % i == 0 THEN
    flag = 1
END FOR
IF flag == 0 THEN
  PRINT "Prime"
ELSE
  PRINT "Not Prime"
```

3. Flowchart of number is prime:



Q10. Define function. Explain user-defined and built-in functions with example. Differentiate call by value and call by reference.

Function:

A block of code that performs a specific task and can be reused. It improves modularity and code reusability.

1. Built-in Functions:

   Already defined in C libraries.

   Example:

```
printf(), scanf(), strlen(), pow()
```

2. User-defined Functions:

   Created by the programmer.

   Syntax:

```
return_type function_name(parameters) {
    // code
}
```

Example:

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

3. Call by Value:

   Actual value is passed.

   Changes made inside function do not affect original.

Example:

```
void change(int x) {
    x = x + 10;
}
```

4. Call by Reference:

   Address is passed.

   Changes made inside function affect the original.

Example:

```
void change(int *x) {
    *x = *x + 10;
}
```

Q11. What is a pointer? Write a C program to read a set of strings and sort them in alphabetical order. Explain declaration and initialization of array of strings.

Pointer:

A pointer is a variable that stores the address of another variable.

Syntax:

```
int a = 5;
int *p = &a;
```

Array of Strings:

An array where each element is a string (character array).

Declaration:

```
char str[5][20];  // 5 strings, each with max 19 characters
```

Program to Sort Strings Alphabetically:

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[5][20], temp[20];
    int i, j;

    printf("Enter 5 strings:\n");
    for (i = 0; i < 5; i++) {
        scanf("%s", str[i]);
    }

    for (i = 0; i < 4; i++) {
        for (j = i + 1; j < 5; j++) {
            if (strcmp(str[i], str[j]) > 0) {
                strcpy(temp, str[i]);
                strcpy(str[i], str[j]);
                strcpy(str[j], temp);
            }
        }
    }

    printf("Strings in alphabetical order:\n");
    for (i = 0; i < 5; i++) {
        printf("%s\n", str[i]);
    }

    return 0;
}
```