01. Explain array.

An array in C programming is a way to store multiple values of the same type, like a list of numbers, all in one place. It is declared by giving the data type and the size of the array, like `int numbers[5];` for an array of five integers. The values in an array are accessed using an index, which starts from 0. For example, `numbers[0]=10;` means the first value in the array is 10. Arrays are stored next to each other in memory, making it fast and easy to get a value using its index. C allows one-dimensional arrays, which are like a simple list, and multi-dimensional arrays, like 2D arrays that act as grids or tables. While arrays are useful for handling large amounts of similar data, they have limits, such as a fixed size that can't change once created and the need for all values to be of the same type. Arrays are commonly used because they are simple and efficient. For example, in the code `int numbers[3] = {1, 2, 3};`, an array of three integers is created, where the first value, `numbers[0]`, is 1.

Example:

```c
#include <stdio.h>

int main() {
    int numbers[5] = {10, 20, 30, 40, 50}; // Declaring and ini

    // Accessing and printing each element using a loop
    for (int i = 0; i < 5; i++) {
        printf("Element at index %d: %d\n", i, numbers[i]);
    }

    return 0;
}
```

02. Explain string functions in C.

In C programming, strings are arrays of characters that end with a special null character \0 to mark the end of the string. C provides several built-in functions to manipulate and handle strings, which are included in the string.h library. These functions help with tasks like copying, comparing, and concatenating strings. One common string function is strlen(), which calculates the length of a string, excluding the null character. For example, strlen("Hello") would return 5.

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

int main() {
    char myString[] = "Programming in C";
    printf("Length of the string is: %lu\n", strlen(myString));
    return 0;
}
```

03. Define a structure type, struct personal that would contain the name of a person, date of joining, and salary. Using this structure, write a program to read this information for one person from the keyboard and print the same on the screen.

```c
#include <stdio.h>

struct person {
    char name[50];
    char date_of_joining[15];
    float salary;
};

int main() {
    struct person p;

    printf("Enter name: ");
    fgets(p.name, sizeof(p.name), stdin);

    printf("Enter date of joining (DD/MM/YYYY): ");
    fgets(p.date_of_joining, sizeof(p.date_of_joining), stdin);

    printf("Enter salary: ");
    scanf("%f", &p.salary);
```

```c
int main() {
    struct person p;

    printf("Enter name: ");
    fgets(p.name, sizeof(p.name), stdin);

    printf("Enter date of joining (DD/MM/YYYY): ");
    fgets(p.date_of_joining, sizeof(p.date_of_joining), stdin);

    printf("Enter salary: ");
    scanf("%f", &p.salary);

    printf("\nPerson Details:\n");
    printf("Name: %s", p.name);
    printf("Date of Joining: %s", p.date_of_joining);
    printf("Salary: %.2f\n", p.salary);
}
```

04. Write a C program using a user-defined function to add six double numbers.

```c
#include <stdio.h>

// User-defined function to find the minimum of three integers
int findMinimum(int a, int b, int c) {
    int min = a;
    if (b < min) {
        min = b;
    }
    if (c < min) {
        min = c;
    }
    return min;
}

int main() {
    int num1, num2, num3, min;

    printf("Enter three integers:\n");
```

```c
    }
    if (c < min) {
        min = c;
    }
    return min;
}

int main() {
    int num1, num2, num3, min;

    printf("Enter three integers:\n");
    scanf("%d %d %d", &num1, &num2, &num3);

    min = findMinimum(num1, num2, num3);

    printf("The minimum value is: %d\n", min);
}
```

05. Write a C program using a user-defined function to find the minimum among three integers.

```c
#include <stdio.h>

// User-defined function to find the minimum of three integers
int findMinimum(int a, int b, int c) {
    int min = a;
    if (b < min) {
        min = b;
    }
    if (c < min) {
        min = c;
    }
    return min;
}

int main() {
    int num1, num2, num3, min;

    printf("Enter three integers:\n");
```

```c
    }
    if (c < min) {
        min = c;
    }
    return min;
}

int main() {
    int num1, num2, num3, min;

    printf("Enter three integers:\n");
    scanf("%d %d %d", &num1, &num2, &num3);

    min = findMinimum(num1, num2, num3);

    printf("The minimum value is: %d\n", min);
}
```

06. Write a C program to take input of an integer 'n.' Then determine whether the number is a "Perfect number" or not. Also, determine whether the number is a "Prime number or not.

```c
#include <stdio.h>
#include <stdbool.h>

// Function to check if a number is a perfect number
bool isPerfect(int n) {
    int sum = 0;
    for (int i = 1; i <= n / 2; i++) {
        if (n % i == 0) {
            sum += i;
        }
    }
    return sum == n;
}

// Function to check if a number is a prime number
bool isPrime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
```

```c
// Function to check if a number is a prime number
bool isPrime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if (n % 2 == 0 || n % 3 == 0) return false;
    for (int i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0) return false;
    }
    return true;
}

int main() {
    int n;

    printf("Enter an integer: ");
    scanf("%d", &n);

    if (isPerfect(n)) {
        printf("%d is a Perfect number \n", n);
```

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

    if (isPerfect(n)) {
        printf("%d is a Perfect number.\n", n);
    } else {
        printf("%d is not a Perfect number.\n", n);
    }

    if (isPrime(n)) {
        printf("%d is a Prime number.\n", n);
    } else {
        printf("%d is not a Prime number.\n", n);
    }

    return 0;
}
```