NOTE BOLD: DO NOT COPY THE TEXT HIGHLIGHTED IN YELLOW COLOR.

DATE: LAB REPORT NO.: **1** SET: **C**
TITLE OF THE PROGRAM: **USER-DEFINED FUNCTIONS IN C**
PASSING ARRAY TO UDF

OBJECTIVES

I. To understand the concept of user-defined functions in C programming.
II. To practice implementing user-defined functions for common array operations.
III. To calculate the sum of elements, sort the elements in ascending order, and find the smallest element in an array.

REQUIREMENTS

1. C Compiler (e.g., GCC)
2. Computer System
3. IDE or Text Editor
4. OS compatible with the software

THEORY

User-defined functions in C programming allow us to modularize our code by breaking it into smaller, reusable functions. These functions can be created to perform specific tasks and can be called from the main program as needed. In this project, we will create three user-defined functions to perform various operations on arrays.

1. **WAP to calculate the sum of elements in an array.**

The function `` `int sum(int arr[], int);` `` takes an array as a parameter and calculates the sum of its elements. It iterates through each element of the array, adds them up, and returns the final sum.

2. **WAP to sort the elements of an array in ascending order.**

The function `` `void ascending(int arr[], int size)` `` sorts the elements of the array in ascending order using the bubble sort algorithm. It compares adjacent elements and swaps them if they are in the wrong order until the entire array is sorted.

3. **WAP to find the smallest element in an array.**

The function `` `int smallest(int arr[])` `` finds the smallest element in the array. It initializes a variable with the first element of the array and then compares it with each subsequent element. If a smaller element is found, it updates the variable. Finally, it returns the smallest element.

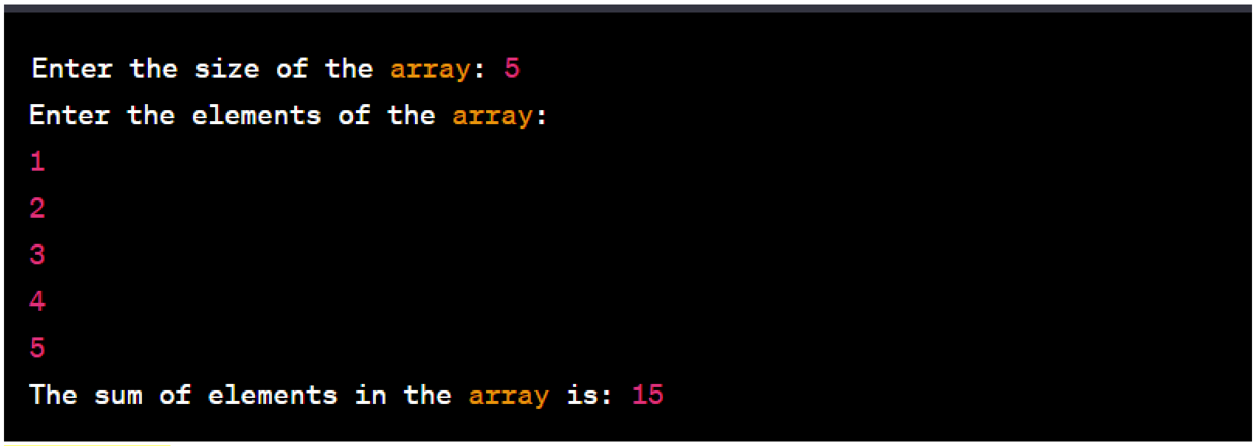
PROCEDURE (Program Code, Comment, and Output)

1. **Calculate the sum of elements in an array.**

Program Code:

```c
#include <stdio.h>

// Function to calculate the sum of elements in an array
int sum(int arr[], int);

int main()
{
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int arr[size];
    printf("Enter the elements of the array:\n");

    // Inputting array elements from the user
    for (int i = 0; i < size; i++)
    {
        scanf("%d", &arr[i]);
    }

    // Calculating the sum of elements in the array
    int arraySum = sum(arr, size);

    // Printing the sum of elements
    printf("The sum of elements in the array is: %d\n", arraySum);

    return 0;
}

// Function definition to calculate the sum of elements in an array
int sum(int arr[], int size)
{
    int sum = 0;

    // Iterating through the array and adding each element to the sum
    for (int i = 0; i < size; i++)
    {
        sum += arr[i];
    }

    // Returning the calculated sum
    return sum;
}
```

Output:



```
Enter the size of the array: 5
Enter the elements of the array:
1
2
3
4
5
The sum of elements in the array is: 15
```

Explanation:

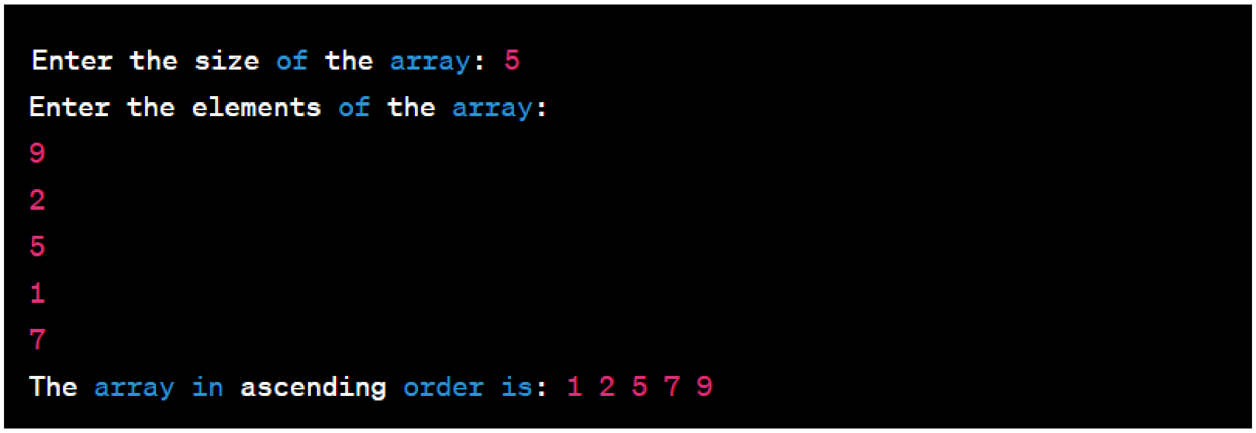
The program starts by including the necessary header files and declaring the user-defined function `sum()`. In the `sum()` function, an integer variable `sum` is initialized to 0 to store the sum of elements. A `for` loop is used to iterate over each element of the array. The current element is added to the `sum` variable in each iteration. After the loop, the `sum` value is returned. In the `main()` function, the user is prompted to enter the size of the array and the elements of the array. The `sum()` function is called with the array and size as arguments, and the returned value is stored in the `arraySum` variable. Finally, the sum of elements is displayed to the user.

2. **Sort the elements of an array in ascending order.**

Program Code:

```c
#include <stdio.h>

// Function to sort the array in ascending order
void ascending(int arr[], int);

int main()
{
    int size, i;
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int arr[size];
    printf("Enter the elements of the array:\n");
    for (i = 0; i < size; i++)
    {
        scanf("%d", &arr[i]);
    }
    ascending(arr, size);  // Call the ascending function to sort the
array

    return 0;
}

void ascending(int arr[], int size)
{
    int i, j, temp;

    // Bubble sort algorithm to sort the array
    for (i = 0; i < size - 1; i++)
    {
        for (j = 0; j < size - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])  // Compare adjacent elements
            {
                temp = arr[j];  // Swap the elements
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    printf("The array in ascending order is: ");
    for (i = 0; i < size; i++)
    {
        printf("%d ", arr[i]);  // Print the sorted array
    }
    printf("\n");
}
```

Output:



```
Enter the size of the array: 5
Enter the elements of the array:
9
2
5
1
7
The array in ascending order is: 1 2 5 7 9
```

Explanation:

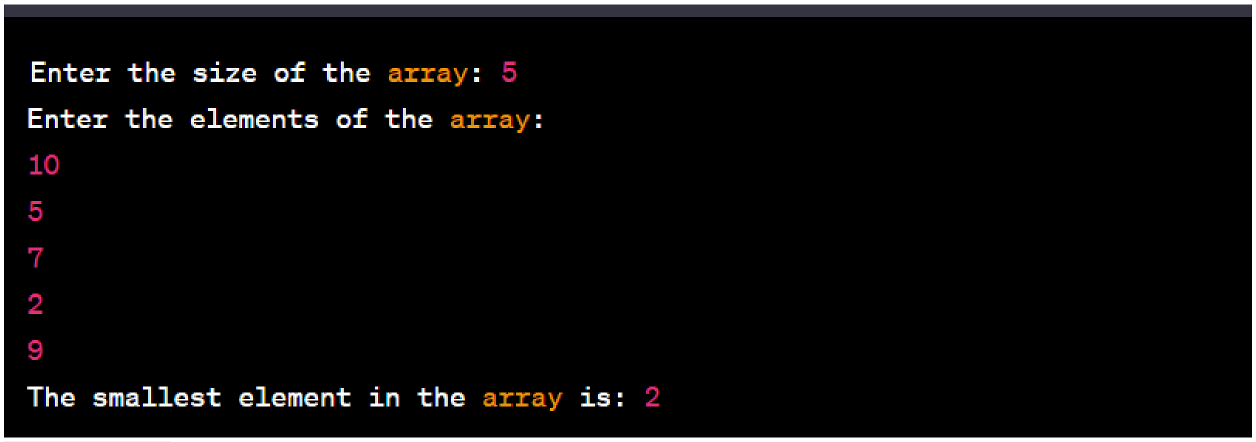
The program starts by including the necessary header files and declaring the user-defined function `ascending()`. In the `ascending()` function, two nested `for` loops are used to implement the Bubble Sort algorithm. The algorithm compares adjacent elements and swaps them if they are in the wrong order. After sorting, the sorted array is displayed using a `for` loop. In the `main()` function, the user is prompted to enter the size of the array and the elements of the array. The `ascending()` function is called with the array and size as arguments to sort and display the sorted array. Finally, the sorted array in ascending order is shown to the user.

3. **Find the smallest element in an array.**

Program Code:

```
#include <stdio.h>

// Function to find the smallest element in an array
int smallest(int arr[], int);

int main()
{
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int arr[size];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++)
    {
        scanf("%d", &arr[i]);
    }

    int smallestElement = smallest(arr, size);
    printf("The smallest element in the array is: %d\n",
smallestElement);

    return 0;
}

// Function to find the smallest element in an array
int smallest(int arr[], int size)
{
    // Assume the first element is the smallest
    int smallest = arr[0];

    // Iterate through the array to find the smallest element
    for (int i = 1; i < size; i++)
    {
        // If a smaller element is found, update the smallest variable
        if (arr[i] < smallest)
        {
            smallest = arr[i];
        }
    }

    // Return the smallest element
    return smallest;
}
```

Output:



```
Enter the size of the array: 5
Enter the elements of the array:
10
5
7
2
9
The smallest element in the array is: 2
```

Explanation:

The program starts by including the necessary header files and declaring the user-defined function `smallest()`. In the `smallest()` function, an integer variable `smallest` is initialized with the value of the first element in the array. A `for` loop is used to iterate over each element of the array, starting from the second element. Each element is compared to the value stored in the `smallest` variable inside the loop. If a smaller element is found, the value of `smallest` is updated. After the loop, the value of the smallest element is returned. In the `main()` function, the user is prompted to enter the size of the array and the elements of the array. The `smallest()` function is called with the array and size as arguments, and the returned value is stored in the `smallestElement` variable. Finally, the smallest element in the array is displayed to the user.

CONCLUSION

Through the implementation of user-defined functions in C programming, this project reinforced modular programming and code reusability. The practical applications of these functions in solving array-related problems, such as sum calculation, array sorting, and finding the smallest element, deepened our understanding of their concepts.