CS1160
Lab 8: Functions

What is a Function?

A function is a group of statements that together perform a specific task. Every C program has at least one function, which is main.

Types of functions:

1. **Standard library functions**: built-in functions in C are declared in header files. such as:
    a. stdio.h : scanf(x), printf(x)

2. **User-defined functions**

Function Declaration & Definition

A function declaration tells the compiler the function's name, return type, and the parameters it needs. Then the function definition provide the actual body of the function.

```
return_type   function_name( parameters list ) {

  //body of the function

}
```

- **Return Type** – the data type of the value the function will return. Some functions perform operations without returning a value. In this case, the **return_type is "void".**
- **Function Name** – the function name and the parameter list together constitute the function signature.
- **Parameters List** – the type, order, and number of the parameters that the function need. Parameters are optional; a function may need no parameters.
- **Function Body** – A collection of statements that define what the function does.

Calling a Function

```
functionName (argument1, argument2, ...);
```

To call a function, simply write the function name and pass the required parameters. If the function returns a value, store the returned value in a variable.

1. **Call by Value:**
   This method copies the actual value of a variable into a function parameter. The changes made to the parameter inside the function do not affect the original variable.
   Example: sum (x)
2. **Call by reference:**
   This method copies the address of a variable to function parameter; where the address is used to access the actual variable. This means that changes made to the parameter inside the function do affect the original variable. Example: array passed to a function.

Example 1

This program calculate the sum of array elements by passing it to a function.

```
1.  #include <stdio.h>
2.  // Function declaration
3.  float SumFunc( float age[ ], int x ) { //function receives an array as a parameter
4.  float sum = 0.0;
5.  int i;
6.  for ( i = 0; i < x;  i++) {
7.          sum += age[i];
8.    }
9.    return sum;
10. }
11.
12. int main() {
13.    float age[] = {23.4, 55, 22.6, 3, 40.5, 18};
14.    float result = SumFunc (age, 6);  // Function call, array is passed to a function
15.    printf (" Result = %.2f ", result);
16.    return 0;
17. }
```

In this example, the **array is passed by reference**. This is because the function receives a reference (memory address) to the first element of the array, and any modifications made to the array within the function will affect the original array in the main function.

Example 2

```
#include <stdio.h>
// Function declaration
void Modify ( float ar[ ]) { //size is passed by reference
int i;
for ( i = 0; i < 5;  i++) {
       ar[i] += 2; }
printf ("\n Array elements in the Modify-fun:\n);

for ( i = 0; i < size;  i++) {
       printf ("%0.2f \t", ar[i]); }
}

int main() {
 int size = 5;
   int ar[size] = {4, 5, 2, 3, 0};
   int i;
  printf ("\n Array elements before the call to Modify-fun:\n);

   for ( i = 0; i < size;  i++) {
              printf ("%0.2f \t", ar[i]); }

Modify(ar, 5);

    printf ("\n Array elements AFTER the call to Modify-fun:\n);
   for ( i = 0; i < size;  i++) {
              printf ("%0.2f \t", ar[i]);  }
return 0; }
```

Examples

1. Write a C program that does the following:

   - Declare and reads three float values.
   - Write a function that receives the integers and calculate their average using the following prototype:

     - float average (float a, float b , float c);

```c
#include <stdio.h>

/* Function prototype */
float average(float a, float b, float c);

int main() {
    float x, y, z, avg;

    /* Read three float values */
    printf("Enter three float numbers: ");
    scanf("%f %f %f", &x, &y, &z);

    /* Call the function */
    avg = average(x, y, z);

    /* Print the result */
    printf("Average = %.2f\n", avg);

    return 0;
}

/* Function definition */
float average(float a, float b, float c) {
    return (a + b + c) / 3;
}
```

```
Enter three float numbers: 80 96 97
Average = 91.00
```

2. Write a c program that will compare two arrays of floats each of size 10**. The comparison process must be done in a separate function.**

```c
#include <stdio.h>
#include <stdbool.h>
/* Function prototype */
bool compareArrays(float arr1[], float arr2[], int size);
int main() {
    float array1[10], array2[10];
    /* Read first array */
    printf("Enter 10 float numbers for the first array:\n");
    for (int i = 0; i < 10; i++) {
        scanf("%f", &array1[i]);
    }

    /* Read second array */
    printf("Enter 10 float numbers for the second array:\n");
    for (int i = 0; i < 10; i++) {
        scanf("%f", &array2[i]);
    }

    /* Compare arrays */
    if (compareArrays(array1, array2, 10)) {
        printf("The two arrays are identical.\n");
    } else {
        printf("The two arrays are NOT identical.\n");
    }
    return 0;
}
/* Function definition */
bool compareArrays(float arr1[], float arr2[], int size) {
    for (int i = 0; i < size; i++) {
        if (arr1[i] != arr2[i]) {
            return false;  // Arrays differ
        }
    }
    return true;  // Arrays are identical
}
```

Tasks

1. Write a C program that does the following:

a. Declare and read **four float values** representing the **lengths of the sides of a rectangle and a triangle**.
b. Write a function that receives the **four float numbers** and calculates:
    1. The **average of all four numbers**
    2. The **sum of the first two numbers** (rectangle sides area approximation)
    3. The **perimeter of the triangle** (last three numbers)

Sample Output:

```
Enter four float numbers: 5.0 3.0 4.0 6.0

Average of all four numbers: 4.50
Rectangle area (first two numbers as length and width): 15.00
Triangle perimeter (last three numbers): 13.00
```

2. Write a C program that does the following:

a. Declare **two arrays of 8 float numbers each** representing the **temperatures recorded in two different cities**.
b. Write a **separate function** that receives the two arrays and calculates:
c. The **highest temperature in each city**
d. The **difference in average temperature between the two cities**.

Sample Output:

```
Enter temperatures for City 1 (8 values): 25.5 27.0 30.2 28.1 26.3 29.5 31.0 27.8
Enter temperatures for City 2 (8 values): 24.0 26.5 28.0 29.0 27.5 30.0 28.8 26.9

Maximum temperature in City 1: 31.00
Maximum temperature in City 2: 30.00
Average temperature difference between City 1 and City 2: 1.23
```