CS1160
Lab 9: Functions 2

Local, Global & Static Variables

- **Local variables**: declared inside a function and can be only accessed within this function.
- **Global variables:** declared outside of all functions, so they are accessible anywhere in the program.
- **Static variables**: initialized only once, if not, the default value is zero. The compiler maintains the variable until the end of the program run. Static variables can be defined inside or outside the function, but they are local to the block they are defined in.

  Example:

```
#include <stdio.h>
// Global variable: Declared outside any function, accessible throughout the program
int globalVar = 10;

void exampleFunction() {

  // Local variable: Declared inside the function, only accessible within it
    int localVar = 20;

    // Static variable: Retains its value across function calls
    static int staticVar = 30;  // Static variables are initialized only once
    printf("In exampleFunction() - Local Variable: %d\n", localVar);
    printf("In exampleFunction() - Static Variable: %d\n", staticVar);
    staticVar++;  // Increment staticVar to demonstrate persistence across calls
}
int main() {
    printf("In main() - Global Variable: %d\n", globalVar);
   // Call exampleFunction() multiple times
    exampleFunction();
    exampleFunction();
        return 0;
}
```

Recursion functions

- Recursion is the technique of making a function call itself.
- This technique breaks complicated problems down into simpler problems easier to solve.
- It keeps calling itself until it reaches a base case.

Example Using Recursion: The Fibonacci sequence

The Fibonacci sequence is a sequence in which each number is the sum of the two preceding ones.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 ...

Example Code for a Fibonacci function:

```
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

Examples

1. Write a C program that does the following:

   - Reads an integer value.
   - Write a function that receives the integer and check whether the number is prime or not.

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

/* Function prototype */
bool isPrime(int n);
int main() {
    int num;

    /* Read an integer */
    printf("Enter an integer: ");
    scanf("%d", &num);

    /* Check if the number is prime */
    if (isPrime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is NOT a prime number.\n", num);
    }

    return 0;
}
/* Function definition */
bool isPrime(int n) {
    if (n <= 1) {
        return false; // 0 and 1 are not prime
    }

    for (int i = 2; i * i <= n; i++) { // check divisibility up to sqrt(n)
        if (n % i == 0) {
            return false; // divisible by i → not prime
        }
    }
    return true; // no divisors found → prime
}

```

2. Write a recursive C function to calculate the factorial of a given non-negative integer `n`. The factorial of a number `n`, denoted as `n!`, is the product of all positive integers less than or equal to `n`.

```c
#include <stdio.h>

/* Function prototype */
int factorial(int n);

int main() {
    int num;
    int result;

    /* Read a non-negative integer */
    printf("Enter a non-negative integer: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        result = factorial(num);
        printf("%d! = %d\n", num, result);
    }

    return 0;
}

/* Recursive function definition */
int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1; // Base case
    } else {
        return n * factorial(n - 1); // Recursive case
    }
}
```

Sample Output:

```
Enter a non-negative integer: 6

6! = 720
```

3. Write a C program that computes and prints the inner product of two integer vectors (one-dimensional arrays) each of size 5. Inner product can be performed by multiplying each element of one vector by the corresponding element in the second vector that has the same index, and summing all the products. Your program should include:
   a. A function of name **read** that scans the elements of each vector.

   b. A function of name **multiply** to perform the inner product and returns the result. Functions above should be called from within the main() function.

Solution:

```c
#include <stdio.h>

/* Function prototypes */
void readVector(int vec[], int size, const char *name);
int multiplyVectors(int vec1[], int vec2[], int size);

int main() {
    int vector1[5], vector2[5];
    int innerProduct;

    /* Read both vectors */
    readVector(vector1, 5, "Vector 1");
    readVector(vector2, 5, "Vector 2");

    /* Compute inner product */
    innerProduct = multiplyVectors(vector1, vector2, 5);

    /* Print the result */
    printf("The inner product of the two vectors is: %d\n", innerProduct);

    return 0;
}

/* Function to read a vector */
void readVector(int vec[], int size, const char *name) {
    printf("Enter %d integers for %s:\n", size, name);
    for (int i = 0; i < size; i++) {
        scanf("%d", &vec[i]);
    }
```

```c
/* Function to compute inner product */
int multiplyVectors(int vec1[], int vec2[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += vec1[i] * vec2[i];
    }
    return sum;
}
```

```
Enter 5 integers for Vector 1:
 1 2 3 4 5 6 7 8
Enter 5 integers for Vector 2:
 3 4 5 6 7
The inner product of the two vectors is: 76
```

Tasks

1. **Write a C program that does the following:**

a. Reads an **integer value** from the user.

b. Write a **function** that receives the integer and checks whether the number is a **Fibonacci number**.

c. A **Fibonacci number** is a number that appears in the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, …

Sample Output:

```
Enter an integer: 8
8 is a Fibonacci number.
```

```
Enter an integer: 7
7 is NOT a Fibonacci number.
```

2. **Write a C program that does the following:**

a. Reads a **non-negative integer n** from the user.

b. Write a **recursive function** that **calculates the sum of all digits** of the number n.

Sample Output:

```
Enter a non-negative integer: 1234
Sum of digits = 10
```

```
Enter a non-negative integer: 507
Sum of digits = 12
```

3. **Write a C program that does the following:**

a. Reads **two arrays of 6 integers each**, representing the **daily sales of two stores over a week (6 days)**.

b. Write a **function** named **readSales** to scan the sales of each store.

c. Write a **function** named **totalSales** that receives the two arrays and **computes the total combined sales for each day** (sum of corresponding elements) and **returns the total sum of all days**.

```
void readSales(int sales[], int size, const char *storeName);
int totalSales(int store1[], int store2[], int size);
```

Sample Output:

```
Enter daily sales for Store 1 (6 days):
10 12 15 20 18 25
Enter daily sales for Store 2 (6 days):
8 14 16 18 22 20

Total combined sales of all days: 188
```