✅ Day : Functions (8-8-2025)

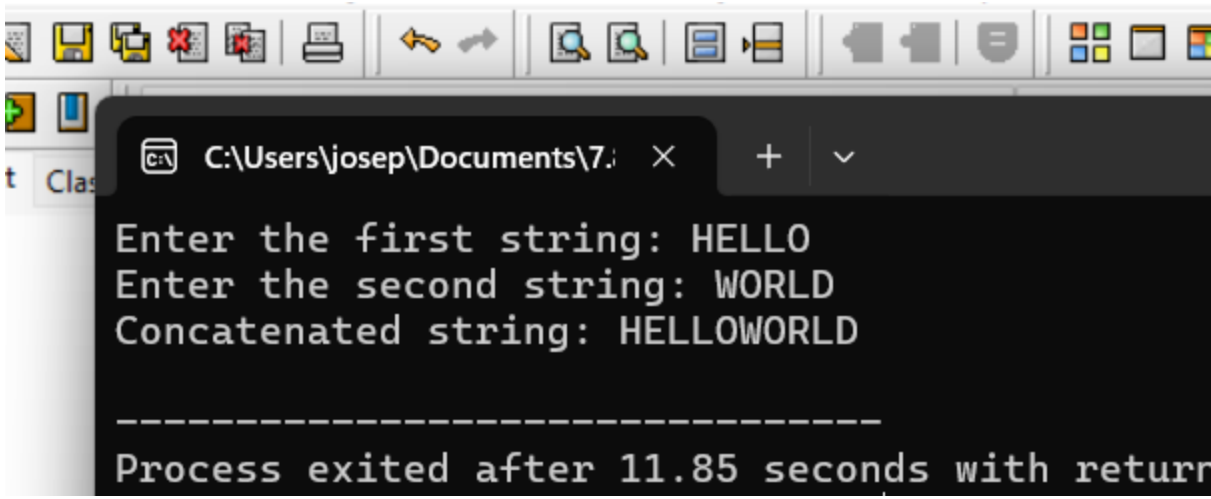
1. Write a function to find the factorial of a number.

Input: A number entered by the user

Process: Multiply all integers from 1 to the given number using a function

Output: Factorial of the number

```
#include <stdio.h>
long long factorial(int n)
{
   long long fact = 1;
   for (int i = 1; i <= n; i++)
{
      fact *= i;
   }
   return fact;
}

int main()
{
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);
   if (num < 0)
      printf("Factorial is not defined for negative numbers.\n");
   else
      printf("Factorial of %d is %lld\n", num, factorial(num));
   return 0;
}
```



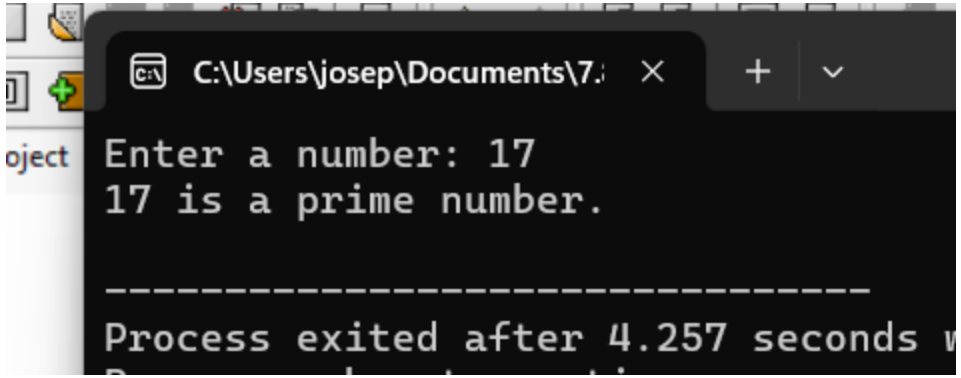
2. Write a function to check whether a number is prime.

Input: A number entered by the user

Process: Check if the number has any divisors other than 1 and itself

Output: Display whether the number is prime or not

```
#include <stdio.h>
int isPrime(int n)
{
   int i;
   if (n <= 1)
      return 0;
   for (i = 2; i <= n / 2; i++) {
      if (n % i == 0)
         return 0;
   }
   return 1;
}
int main()
{
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);
   if (isPrime(num))
      printf("%d is a prime number.\n", num);
   else
      printf("%d is not a prime number.\n", num);
   return 0;
}
```



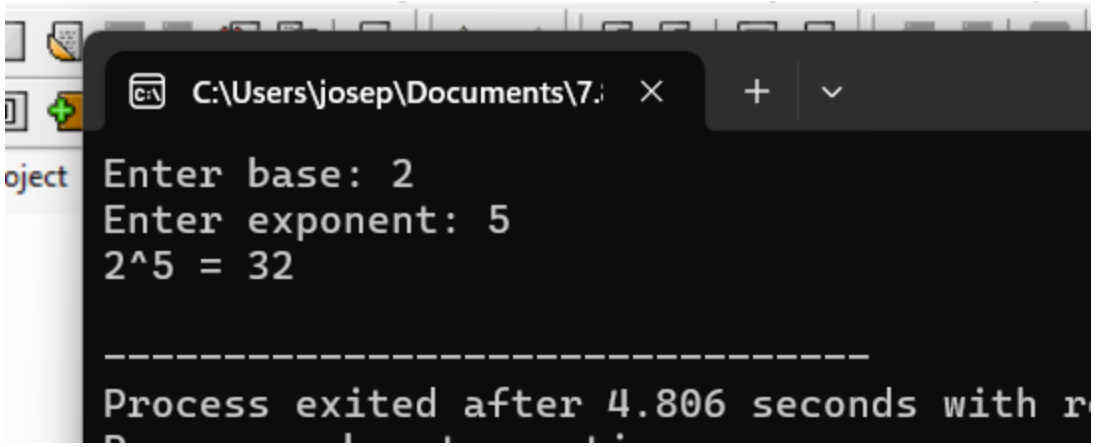
3. Write a function to calculate power using recursion.

Input: Base and exponent entered by the user

Process: Use a recursive function to calculate base^exponent

Output: Display the result of base raised to the power exponent

```
#include <stdio.h>
int power(int base, int exp) {
   if (exp == 0)
      return 1;
   else
      return base * power(base, exp - 1);
}
int main() {
   int base, exponent, result;
   printf("Enter base: ");
   scanf("%d", &base);
   printf("Enter exponent: ");
   scanf("%d", &exponent);
   result = power(base, exponent);
   printf("%d^%d = %d\n", base, exponent, result);
   return 0;
}
```



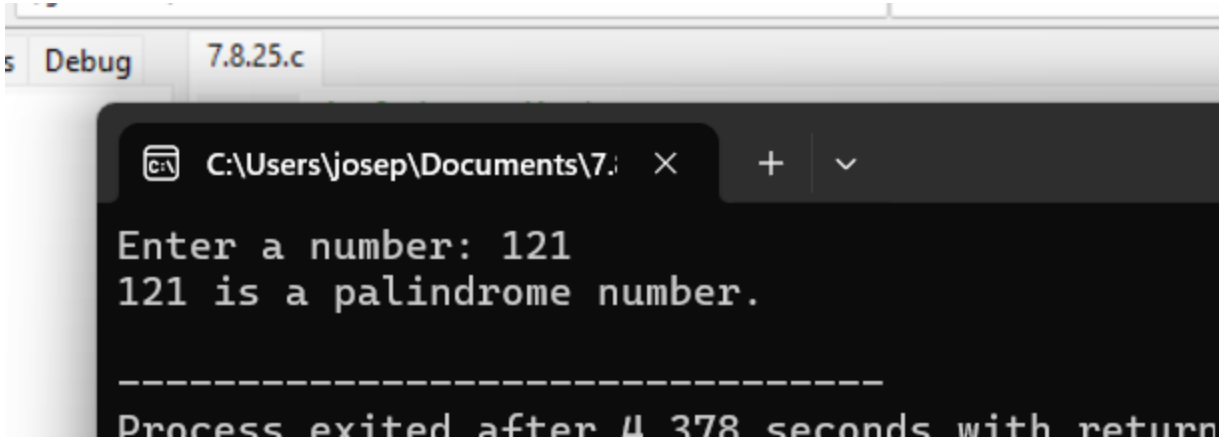
```c
#include <stdio.h>
int reverse(int num, int rev)
{
   if (num == 0)
      return rev;
   rev = rev * 10 + num % 10;
   return reverse(num / 10, rev);
}

int isPalindrome(int num)
{
   int reversed = reverse(num, 0);
   if (num == reversed)
      return 1;
   else
      return 0;
}

int main()
{
   int number;
   printf("Enter a number: ");
   scanf("%d", &number);
   if (isPalindrome(number))
      printf("%d is a palindrome number.\n", number);
   else
      printf("%d is not a palindrome number.\n", number);

   return 0;
}
```

4. Write a function to check palindrome number using recursion.

Input: A number entered by the user

Process: Reverse the number using recursion and compare it with the original

Output: Display whether the number is a palindrome or not



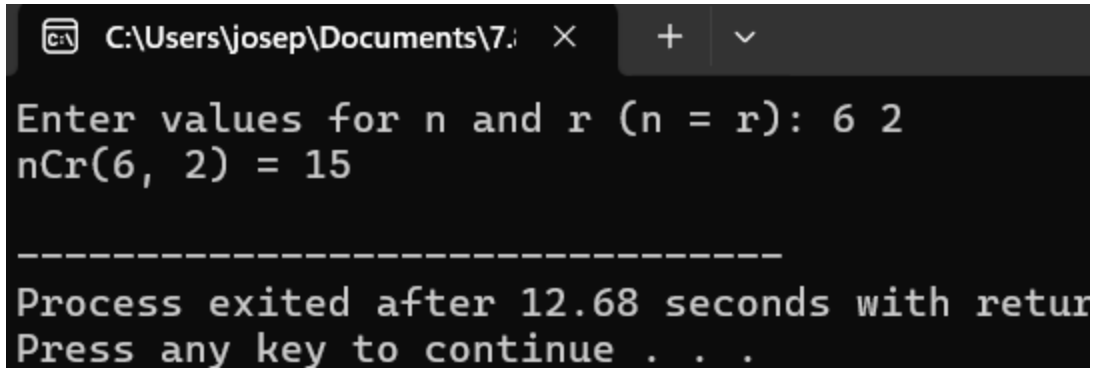
5. Write a function to calculate nCr (combinations).

Input: Two integers n and r

Process: Calculate `n!`, `r!`, and `(n-r)!`, then compute `nCr = n! / (r! * (n - r)!)`

Output: Value of `nCr`

```
#include <stdio.h>
int fact(int n)
{
    int f = 1, i;
    for (i = 1; i <= n; i++)
        f *= i;
    return f;
}
int main()
{
    int n, r, result;
    printf("Enter n and r: ");
    scanf("%d%d", &n, &r);
    if (r > n || n < 0 || r < 0)
        printf("Invalid input\n");
    else
{
        result = fact(n) / (fact(r) * fact(n - r));
        printf("nCr = %d\n", result);
    }
    return 0;
}
```



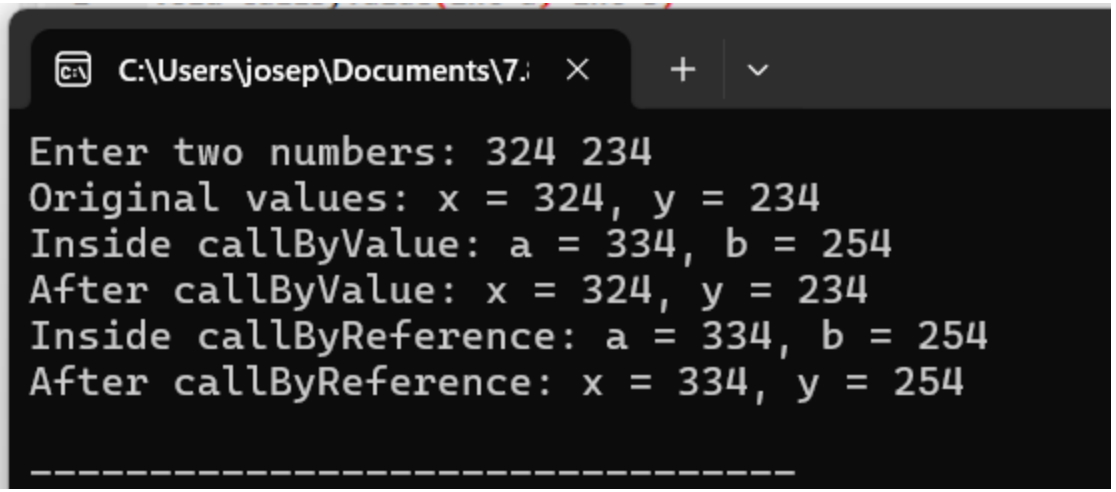
6. Write a program to demonstrate call by value and call by reference.

INPUT:
- → Read two integers x and y from the user.

PROCESS:
- → Pass x and y to callByValue() – does NOT change original.
- → Pass address of x and y to callByReference() – modifies original.

OUTPUT:
- → Show original values.
- → Show values after callByValue (unchanged).
- → Show values after callByReference (changed)

```
#include <stdio.h>
void callByValue(int a, int b) { a = a + 10; b = b + 20;
 printf("\nInside callByValue: a = %d, b = %d", a, b);
}
void callByReference(int *a, int *b)
{
*a = *a + 10; *b = *b + 20;
printf("\nInside callByReference: a = %d, b = %d", *a, *b);
}
int main()
 {
int x = 5, y = 10;
printf("Original values: x = %d, y = %d", x, y);
callByValue(x, y);
printf("\nAfter callByValue: x = %d, y = %d", x, y);
callByReference(&x, &y);
printf("\nAfter callByReference: x = %d, y = %d\n", x, y);
return 0;
}
```



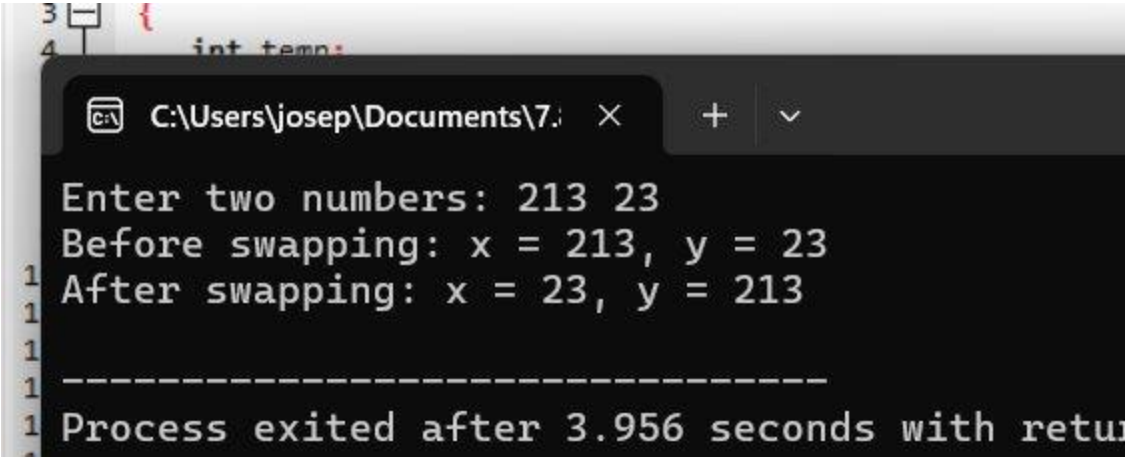
7. Write a program using function to swap two numbers.

INPUT:
- → User enters two numbers (x and y)

PROCESS:
- → Use swap() function with pointers to swap values of x and y

OUTPUT:
- → Print x and y before swap
- → Print x and y after swap

```
#include <stdio.h>
 void swap(int *a, int *b)
 {
int temp; temp = *a; *a = *b; *b = temp;
} i
nt main()
 { int x, y; printf("Enter two numbers: ");
scanf("%d %d", &x, &y);
printf("Before swapping: x = %d, y = %d\n", x, y);
 swap(&x, &y);
 printf("After swapping: x = %d, y = %d\n", x, y);
 return 0;
 }
```



8. Write a recursive function to find the nth Fibonacci number.

INPUT:

→ Read integer n (position in Fibonacci sequence)

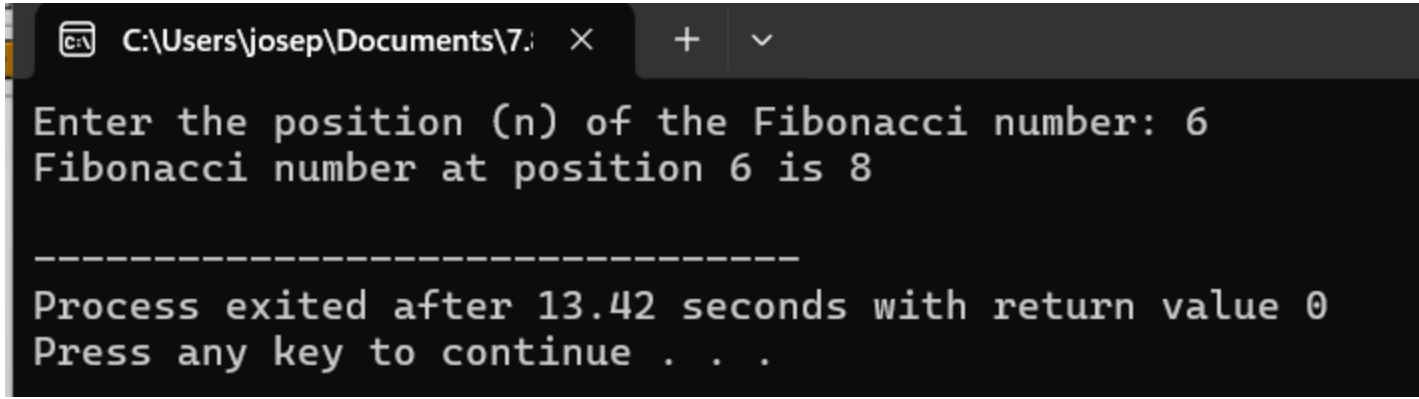
PROCESS:

→ Use recursion to compute the nth Fibonacci number:
fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
with base cases:
fibonacci(0) = 0
fibonacci(1) = 1

OUTPUT:

→ Display the nth Fibonacci number

```
#include <stdio.h>
int fibonacci(int n)
{
   if (n == 0)
      return 0;
   else if (n == 1)
      return 1;
   else
      return fibonacci(n - 1) + fibonacci(n - 2);
}
int main()
{
   int n, result;
   printf("Enter the position (n) of the Fibonacci number: ");
   scanf("%d", &n);
   result = fibonacci(n);
   printf("Fibonacci number at position %d is %d\n", n, result);
   return 0;
}
```



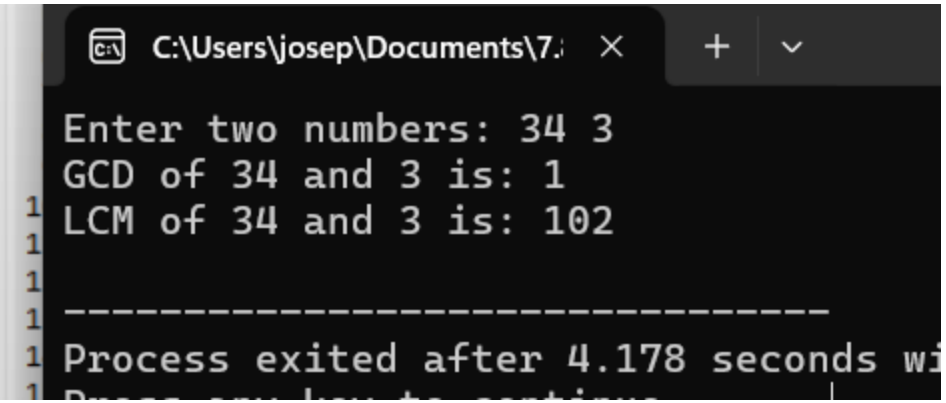
9. Write a program to find GCD and LCM using functions.

INPUT:
- → Read two integers (num1 and num2)

PROCESS:
- → Use findGCD() to calculate greatest common divisor
- → Use findLCM() to calculate least common multiple: (a × b) / GCD

OUTPUT:
- → Display GCD and LCM of the two numbers

```
#include <stdio.h>
int findGCD(int a, int b)
{
   while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
   }
   return a;
}
int findLCM(int a, int b) {
   return (a * b) / findGCD(a, b);
}

int main()
{
   int num1, num2, gcd, lcm;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);
   gcd = findGCD(num1, num2);
   lcm = findLCM(num1, num2);
   printf("GCD of %d and %d is: %d\n", num1, num2, gcd);
   printf("LCM of %d and %d is: %d\n", num1, num2, lcm);
   return 0;
```



10.Write a program to demonstrate global and local variables.

INPUT:
- → No user input needed (we're demonstrating variable scope)

PROCESS:
- → Declare and print a local variable inside a function
- → Access a global variable both inside the function and in main

OUTPUT:
- → Print values of local and global variables
- → Show that global is accessible everywhere, but local is not

```
#include <stdio.h>
int globalVar = 100;
void showVariables() {
   int localVar = 50;
   printf("Inside function:\n");
   printf("Local Variable = %d\n", localVar);
   printf("Global Variable = %d\n", globalVar);
}

int main()
{
   showVariables();
   printf("Inside main:\n");
   printf("Global Variable = %d\n", globalVar);
   return 0;
}
```