

## ASSIGNMENT 6(8.08.2025)

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

**Input:** An integer n

**Process:** Multiply numbers from 1 to n  $\rightarrow n! = 1 \times 2 \times 3 \times \dots \times n$

**Output:** Factorial of n

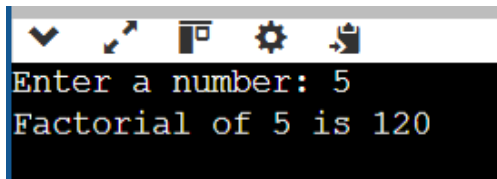
#### CODE:

```
#include <stdio.h>

int factorial(int n)
{
    int fact = 1;
    for(int i = 1; i <= n; i++)
        fact *= i;
    return fact;
}

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Factorial of %d is %d\n", num, factorial(num));
    return 0;
}
```

#### OUTPUT:



```
Enter a number: 5
Factorial of 5 is 120
```

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

**prime Input:** An integer n

**Process:** Check if n is divisible by any number from 2 to  $n/2$

**Output:** Prime or not prime message

**CODE:**

```
#include <stdio.h>

#include <stdbool.h>

bool isPrime(int n)
{
    if(n < 2) return false;

    for(int i = 2; i <= n/2; i++)
        if(n % i == 0)
            return false;

    return true;
}

int main()
{
    int num;
```

```

printf("Enter a number: ");

scanf("%d", &num);

if(isPrime(num))

    printf("%d is a prime number\n", num);

else

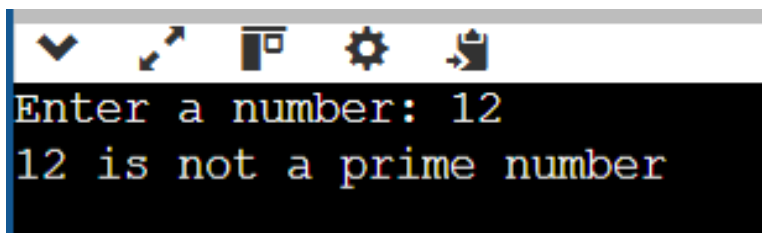
    printf("%d is not a prime number\n", num);

return 0;

}

```

### OUTPUT:



```

Enter a number: 12
12 is not a prime number

```

### 3. Write a function to calculate power using recursion.

**Input:** Base b, Exponent e

**Process:** Recursive multiplication  $\rightarrow b^e = b \times b \times \dots \times b$  (e times)

**Output:** Result of  $b^e$

CODE :

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

```
int power(int base, int exp)
```

```

{
    if(exp == 0)
        return 1;
    return base * power(base, exp - 1);
}

int main()
{
    int base, exp;

    printf("Enter base and exponent: ");

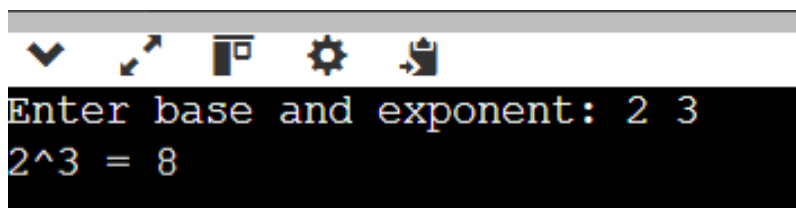
    scanf("%d %d", &base, &exp);

    printf("%d^%d = %d\n", base, exp, power(base, exp));

    return 0;
}

```

OUTPUT:



```

Enter base and exponent: 2 3
2^3 = 8

```

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

**Input:** An integer n

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

**Output:** Whether the number is a palindrome

CODE:

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

```
int reverse(int num, int rev)
```

```
{
```

```
    if(num == 0)
```

```
        return rev;
```

```
    return reverse(num / 10, rev * 10 + num % 10);
```

```
}
```

```
int main()
```

```
{
```

```
    int num, rev;
```

```
    printf("Enter a number: ");
```

```
    scanf("%d", &num);
```

```
    rev = reverse(num, 0);
```

```
    if(num == rev)
```

```
        printf("%d is a palindrome\n", num);
```

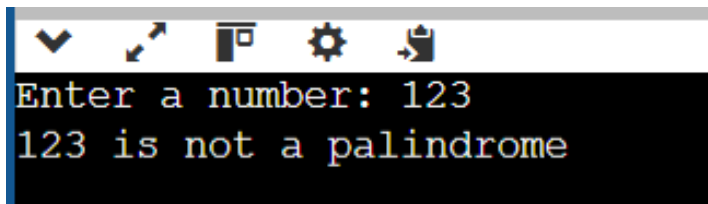
```
    else
```

```
        printf("%d is not a palindrome\n", num);
```

```
    return 0;
```

```
}
```

OUTPUT:



```
Enter a number: 123
123 is not a palindrome
```

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

**Input:** Two integers n and r

**Process:** Compute using formula  $\rightarrow nCr = \frac{n!}{(r! * (n - r)!)}$

**Output:** Value of nCr

CODE:

```
#include <stdio.h>

int factorial(int n) {
    int f = 1;
    for(int i = 1; i <= n; i++)
        f *= i;
    return f;
}

int nCr(int n, int r) {
    return factorial(n) / (factorial(r) * factorial(n - r));
}
```

```

int main() {

    int n, r;

    printf("Enter n and r: ");

    scanf("%d %d", &n, &r);

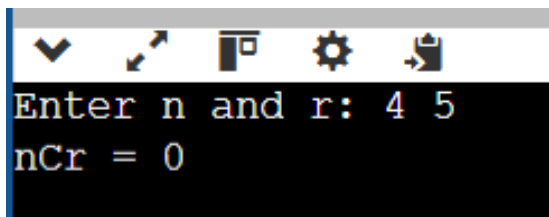
    printf("nCr = %d\n", nCr(n, r));

    return 0;

}

```

OUTPUT:



```

Enter n and r: 4 5
nCr = 0

```

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

**Input:** An integer variable x

**Process:**

- Call by value: Pass x and modify inside function (no effect outside)
- Call by reference: Pass address of x and modify actual value

**Output:** Values before and after both function calls

CODE:

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

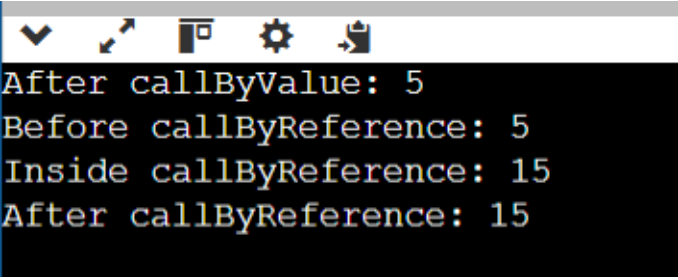
```
void callByValue(int a) {  
    a = a + 10;  
    printf("Inside callByValue: %d\n", a);  
}
```

```
void callByReference(int *a) {  
    *a = *a + 10;  
    printf("Inside callByReference: %d\n", *a);  
}
```

```
int main() {  
    int x = 5;  
    printf("Before callByValue: %d\n", x);  
    callByValue(x);  
    printf("After callByValue: %d\n", x);  
  
    printf("Before callByReference: %d\n", x);  
    callByReference(&x);  
    printf("After callByReference: %d\n", x);  
  
    return 0;  
}
```



OUTPUT:



```
After callByValue: 5
Before callByReference: 5
Inside callByReference: 15
After callByReference: 15
```

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

**Input:** Two numbers a and b

**Process:** Use a temporary variable (or pointers) to swap values

**Output:** Values of a and b after swapping

CODE: #include <stdio.h>

```
void swap(int *a, int *b) {
```

```
    int temp = *a;
```

```
    *a = *b;
```

```
    *b = temp;
```

```
}
```

```
int main()
```

```

{
    int x, y;

    printf("Enter two numbers: ");

    scanf("%d %d", &x, &y);

    printf("Before swap: x = %d, y = %d\n", x, y);

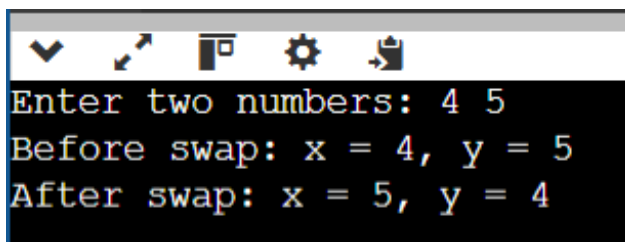
    swap(&x, &y);

    printf("After swap: x = %d, y = %d\n", x, y);

    return 0;
}

```

OUTPUT:



```

Enter two numbers: 4 5
Before swap: x = 4, y = 5
After swap: x = 5, y = 4

```

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

**Input:** Integer n

**Process:** Use recursion to calculate nth Fibonacci number

$F(n) = F(n-1) + F(n-2)$  with base cases  $F(0)=0$ ,  $F(1)=1$

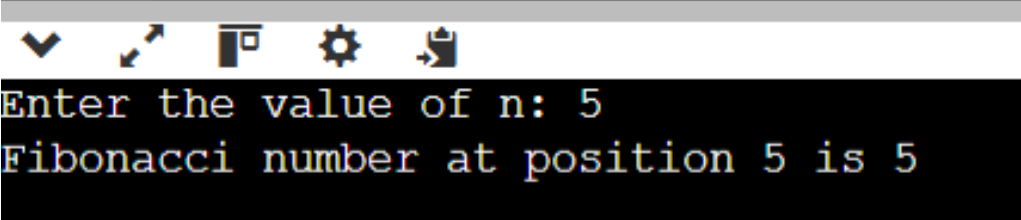
**Output:** nth Fibonacci number

CODE:

```
#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;  
    printf("Enter the value of n: ");  
    scanf("%d", &n);  
    printf("Fibonacci number at position %d is %d\n", n, fibonacci(n));  
    return 0;  
}
```

OUTPUT:



```
Enter the value of n: 5  
Fibonacci number at position 5 is 5
```

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

**Input:** Two integers num1 and num2

**Process:**

a. GCD: Euclidean algorithm

b. LCM:  $(\text{num1} * \text{num2}) / \text{GCD}$

**Output:** GCD and LCM of the input numbers

CODE:

```
#include <stdio.h>

int gcd(int a, int b) {
    while(b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int lcm(int a, int b) {
    return (a * b) / gcd(a, b);
}

int main() {
    int num1, num2;

    printf("Enter two numbers: ");
```

```

scanf("%d %d", &num1, &num2);

printf("GCD = %d\n", gcd(num1, num2));

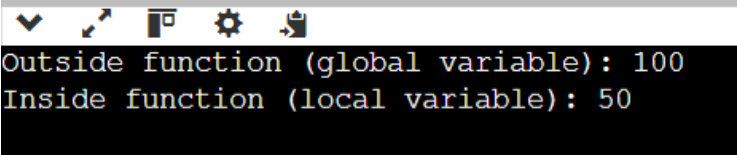
printf("LCM = %d\n", lcm(num1, num2));

return 0;

}

```

OUTPUT:



```

Outside function (global variable): 100
Inside function (local variable): 50

```

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

Input: None (global and local values defined in code)

Process:

- a. Show global variable in main
- b. Shadow global variable with a local one inside show()

Output: Prints both global and local variable values

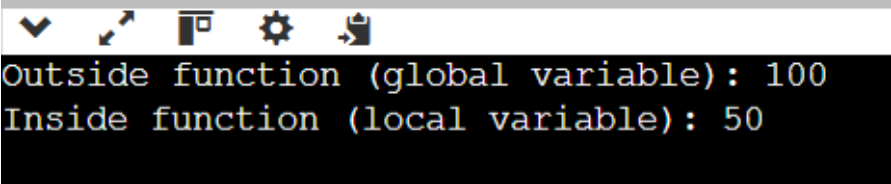
CODE:

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

```
int globalVar = 100; // Global variable
```

```
void show() {  
    int globalVar = 50; // Local variable (same name)  
    printf("Inside function (local variable): %d\n", globalVar);  
}  
  
int main() {  
  
    printf("Outside function (global variable): %d\n", globalVar);  
    show();  
    return 0;  
}
```

OUTPUT:



```
Outside function (global variable): 100  
Inside function (local variable): 50
```