Day : Functions (8-8-2025)

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

IPO:

- **Input:** Integer n
- **Process:** Multiply numbers from 1 to n
- **Output:** Factorial of n

Program

```
#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;
   scanf("%d", &num);
   printf("Factorial = %d\n", factorial(num));
   return 0;
}
```

Output

```
Output
5
Factorial = 120
```

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

IPO:

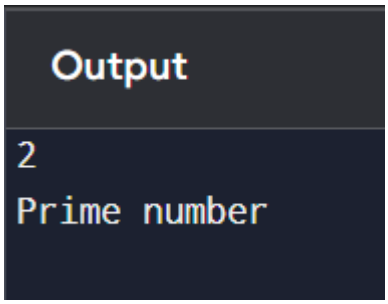
- **Input:** Integer n

- **Process:** Check if divisible by any number from 2 to n-1
- **Output:** Prime or not

Program

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

Output



3. Write a function to calculate power using recursion.

IPO:

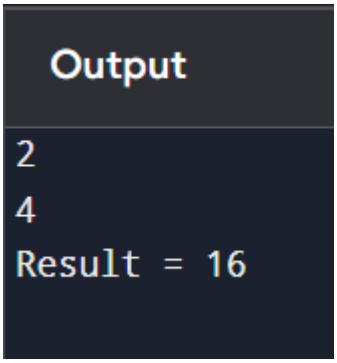
- **Input:** Base x, exponent y

- **Process:** Multiply x recursively y times
- **Output:** Result of x^y

Program

```
#include <stdio.h>
int power(int x, int y)
{
   if (y == 0)
      return 1;
   return x * power(x, y - 1);
}
int main()
{
   int base, exp;
   scanf("%d%d", &base, &exp);
   printf("Result = %d\n", power(base, exp));
   return 0;
}
```

Output



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

IPO:

- **Input:** Number
- **Process:** Reverse number using recursion
- **Output:** Palindrome or not

Program

```
#include <stdio.h>
int reverseNum(int num, int rev)
{
   if (num == 0)
      return rev;
   return reverseNum(num / 10, rev * 10 + num % 10);
}
int main()
{
   int num;
   scanf("%d", &num);
   if (num == reverseNum(num, 0))
      printf("Palindrome\n");
   else
      printf("Not palindrome\n");
   return 0;
}
```

Output

```
Output
121
Palindrome
```

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

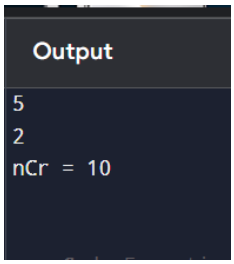
IPO:

- **Input:** n, r
- **Process:** Formula nCr = n! / (r! * (n-r)!)
- **Output:** Value of nCr

Program

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

Output



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

IPO:

- **Input:** Two numbers
- **Process:** Show difference in passing values vs addresses
- **Output:** Values swapped or not

Program

```
#include <stdio.h>
void swapByValue(int m, int n)
{
   int temp = m;
```

```
    m = n;
    n = temp;
}
void swapByReference(int *m, int *n)
{
    int temp = *m;
    *m = *n;
    *n = temp;
}
int main()
{
    int num1 = 15, num2 = 25;
    swapByValue(num1, num2);
    printf("After swap by value: num1=%d, num2=%d\n", num1, num2);
    swapByReference(&num1, &num2);
    printf("After swap by reference: num1=%d, num2=%d\n", num1, num2);
    return 0;
}
```

Output

```
Output
After swap by value: num1=15, num2=25
After swap by reference: num1=25, num2=15
```

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

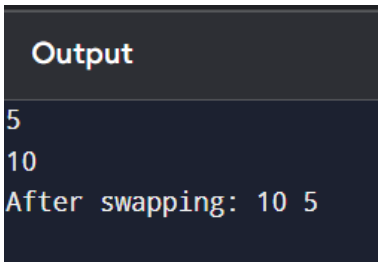
IPO:

- **Input:** Two numbers
- **Process:** Exchange values
- **Output:** Swapped numbers

Program

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

Output



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

IPO:

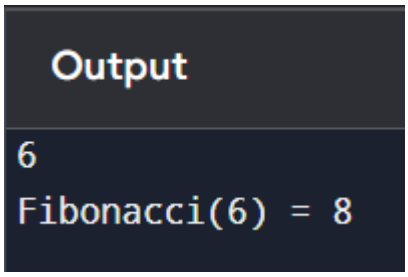
- **Input:** n
- **Process:** Recursively find nth term
- **Output:** nth Fibonacci number

Program

```
#include <stdio.h>
int f(int n)
{
```

```
    if (n <= 1)
        return n;
    return f(n - 1) + f(n - 2);
}
int main()
{
    int n;
    scanf("%d", &n);
    printf("Fibonacci(%d) = %d\n", n, f(n));
    return 0;
}
```

Output



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

IPO:

- **Input:** Two numbers
- **Process:** GCD by Euclid's method, LCM formula = (a*b)/GCD
- **Output:** GCD and LCM

Program

```
#include <stdio.h>
int gcd(int a, int b)
{
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

```
int main()
{
   int a, b;
   scanf("%d%d", &a, &b);
   int g = gcd(a, b);
   int l = (a * b) / g;
   printf("GCD = %d, LCM = %d\n", g, l);
   return 0;
}
```

Output

```
Output
12
8
GCD = 4, LCM = 24
```

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

IPO:

- **Input:** None (values inside program)
- **Process:** Show variable scope
- **Output:** Different values for same variable name

Program

```
#include <stdio.h>
int g = 10;
void display()
{
   int g = 20;
   printf("Local g = %d\n", g);
}
int main()
{
```

```
    printf("Global g = %d\n", g);
    display();
    printf("Global g again = %d\n", g);
    return 0;
}
```

Output

Output

```
Global g = 10
Local g = 20
Global g again = 10
```