1.Check Whether a Given Number is Perfect Number

```
def is_perfect_number(num):
    if num < 1:
        return False

    divisors_sum = sum(i for i in range(1, num) if num % i == 0)
    return divisors_sum == num

# Default number to test
number = 28  # You can change this value to any number you'd like to test

if is_perfect_number(number):
    print(f"{number} is a perfect number.")
else:
    print(f"{number} is not a perfect number.")
```

Expected Output:

28 is a perfect number.

2. Leap year or not:

```
def is_leap_year(year):
    # A leap year is divisible by 4, but if it's a century year, it must also be divisible by 400
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Default year to test
year = 2024  # You can change this value to any year you'd like to test
```

```
# Check if it's a leap year
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Expected Output:

2024 is a leap year.

3. Check whether a Number is Prime or Not using Recursion

```
public class Prime_Check {

    public static void main(String[] args) {
        // Default number to check
        int num = 29; // You can change this value to any number you'd like to check

        if (isPrime(num, num / 2)) {
            System.out.println(num + " is a prime number");
        } else {
            System.out.println(num + " is not a prime number");
        }
    }

    public static boolean isPrime(int num, int i) {
        if (i == 1) {
            return true;
        } else {
            if (num % i == 0) {
                return false;
```

```
        } else {
            return isPrime(num, i - 1);
        }
    }
  }
}
```

Expected Output:

29 is a prime number

4. Pyramid Pattern

```
public class PyramidPattern {
  public static void main(String[] args) {
    // Default number of rows for the pyramid
    int rows = 5; // You can change this value to any number of rows you'd like

    for (int i = 1; i <= rows; i++) {
      // Print spaces
      for (int j = 1; j <= rows - i; j++) {
        System.out.print(" ");
      }

      // Print stars
      for (int k = 1; k <= (2 * i - 1); k++) {
        System.out.print("*");
      }

      // Move to the next line
      System.out.println();
    }
  }
```

```
}
```

Expected Output:

```
    *
   ***
  *****
 *******
*********
```

5. Add Two Binary Numbers

```
#include <stdio.h>

int main() {
   // Default binary numbers
   long binary1 = 1010; // You can change this to any binary number you'd like
   long binary2 = 1101; // You can change this to any binary number you'd like

   int i = 0, remainder = 0, sum[20];

   while (binary1 != 0 || binary2 != 0) {
      sum[i++] = (binary1 % 10 + binary2 % 10 + remainder) % 2;
      remainder = (binary1 % 10 + binary2 % 10 + remainder) / 2;
      binary1 = binary1 / 10;
      binary2 = binary2 / 10;
   }
   if (remainder != 0) {
      sum[i++] = remainder;
   }
   --i;

   printf("Sum of two binary numbers: ");
   while (i >= 0) {
      printf("%d", sum[i--]);
```

```
    }
    return 0;
}
```

Expected Output:

Sum of two binary numbers: 10111

6. Fibonacci Series

```
#include <stdio.h>

int main() {
    // Default number of terms in the Fibonacci series
    int terms = 10; // You can change this value to any number of terms you'd like
    int first = 0, second = 1, next;

    printf("Fibonacci Series up to %d terms:\n", terms);

    for (int i = 1; i <= terms; i++) {
        printf("%d ", first);

        // Calculate the next term
        next = first + second;
        first = second;
        second = next;
    }

    return 0;
}
```

Expected Output:

Fibonacci Series up to 10 terms:

0 1 1 2 3 5 8 13 21 34