Q1. In number theory, a polite number is a positive integer that can be written as the sum of two or more consecutive positive integers. Rest of the integers are said to be impolite.
For e.g. 7 and 18 are polite numbers as 7 = 3+4 and 18=5+6+7 The first few polite numbers are 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, ... The politeness of a positive integer is the number of ways in which we can represent the given number as the sum of positive consecutive integers.
For e.g. 9 has a politeness of 2 as 9 = 2+3+4 as well as 4+5.
Write a function, politeFunc, which takes a positive integer and prints various ways in which it can be represented as a sum of consecutive positive integers.
Each possible representation is printed in a new line.
For e.g.
i. output for polieFunc(10) should be 1 2 3 4
ii. output for polieFunc(10) should be
2 3 4
4 5
If the number is impolite, output the string "I am RUDE!"
Check your function for powers of 2 (2,4,8,16,...).

CODE:

```
/*
Polite numbers: integers expressible as a sum of two or more consecutive positive integers; politeness is the count of such representations.
politeFunc(n) prints each representation on a new line or "I am RUDE!" if none (e.g., powers of 2).

Name: Abhishek Sonkar (2025ca005)
*/


#include <stdio.h>

// Function: politeFunc
// Purpose: Prints all ways n can be written as a sum of consecutive positive integers.
// If n is impolite (cannot be written as such), prints "I am RUDE!"
// Example: For n=10, output is:
//   1+2+3+4
//   2+3+5
// For n=8 (a power of 2), output is:
//   I am RUDE!

// Prints all ways n can be written as a sum of consecutive positive integers.
// If n is impolite, prints "I am RUDE!"
void politeFunc(int n)
{
    int found = 0; // Tracks if at least one representation is found

    // Try all possible lengths of consecutive numbers (at least 2)
    for (int length = 2; length < n; length++)
```

```
    {
        // The sum of 'length' consecutive numbers starting from x is:
        // n = x + (x+1) + ... + (x+length-1) = length*x + length*(length-1)/2
        // Rearranged: x = (n - length*(length-1)/2) / length

        int temp = n - length * (length - 1) / 2;
        if (temp <= 0) break; // No valid starting number x for this length
        if (temp % length == 0)
        {
            int x = temp / length; // Starting number of the sequence
            found = 1;
            // Print the sequence x, x+1, ..., x+length-1 with plus signs
            for (int i = 0; i < length; i++)
            {
                printf("%d", x + i);
                if (i < length - 1) printf("+");
            }
            printf("\n");
        }
    }
    // If no representation found, print the rude message
    if (!found) {
        printf("I am RUDE!\n");
    }
}


// Main function: gets user input and calls politeFunc
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    politeFunc(n);
    return 0;
}
```

```
abhis@Asus_Vivobook MINGW64 ~/CLionProjects/pps-assignment (main)
$ gcc ques1.c

abhis@Asus_Vivobook MINGW64 ~/CLionProjects/pps-assignment (main)
$ ./a.exe
Enter a number: 45
22+23
14+15+16
7+8+9+10+11
5+6+7+8+9+10
1+2+3+4+5+6+7+8+9
```

Q2.Take a number and calculate the sum of the squares of its digits. Create a function that performs this operation a number k times. It is a mathematical fact that starting from any arbitrary positive integer n, if we keep on applying the above operation, we shall eventually get either 1 or 89. For example sqn(17) = 50, sqn(50) = 25, sqn(25) = 29, sqn(29) = 85, sqn(85) = 89 Hence, if you did the operation k=5 times, you would have got 89. Verify that after 5 repetitions on 17, you get 89.

CODE:

```
/*
Calculates and prints the sum of squares of the digits of 'num'.

Name: Abhishek Sonkar (2025ca005)
*/

#include <stdio.h>


// Returns the sum as an integer.
int sumOfSquares(int num) {
    int sum = 0;
    int digits[10], count = 0;

    // Extract digits (in reverse order)
    while (num > 0) {
        digits[count++] = num % 10;
        num /= 10;
    }

    // Print the calculation in readable format
    printf("(");
    for (int i = count - 1; i >= 0; i--) {
        printf("%d^2", digits[i]);
        if (i > 0) printf(" + ");
        sum += digits[i] * digits[i];
    }
    printf(") = %d\n", sum);
    return sum;
}

int main() {
    int n = 17; // Starting number
    int k = 5;  // Number of repetitions
```

```
    printf("Start: %d\n", n);
    for (int step = 1; step <= k; step++) {
        n = sumOfSquares(n);
    }
    printf("Final result after %d steps: %d\n", k, n); // Should print 89
    return 0;
}
```

```
abhis@Asus_Vivobook MINGW64 ~/CLionProjects/pps-assignment (main)
$ gcc ./ques2.c

abhis@Asus_Vivobook MINGW64 ~/CLionProjects/pps-assignment (main)
$ ./a.exe
Start: 17
(1^2 + 7^2) = 50
(5^2 + 0^2) = 25
(2^2 + 5^2) = 29
(2^2 + 9^2) = 85
(8^2 + 5^2) = 89
Final result after 5 steps: 89
```