Assignment 1:

Write a C program that includes a user-defined function named isPrime with the signature int **isPrime(int num);** The function should take an integer as a parameter and return 1 if the number is prime and 0 otherwise.

Source Code :

```
#include <stdio.h>

int isPrime(int num)
{
    int i;

    for (i = 2; i <= num / 2; i++)
    {
        if (num % i == 0)
            return 0;
    }
    return 1;
}

int main()
{
    int n;

    printf("Enter a number: ");
    scanf("%d", &n);

    if (n <= 1)
        printf("%d is not a prime number.", n);
    else if (isPrime(n))
        printf("%d is a prime number.", n);
    else
        printf("%d is not a prime number.", n);
    return 0;
}
```

Output:

Assignment 2:

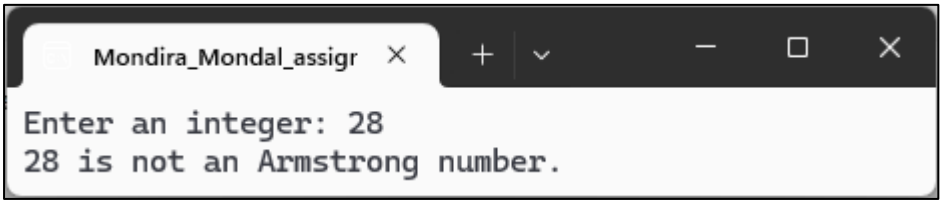
Write a C program that includes a user-defined function named isArmstrong with the signature **int isArmstrong(int num);**. An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits. For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153

Source Code:

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

int isArmstrong(int num)
{
    int originalNum = num;
    int remainder;
    int sum = 0;
    int numberOfDigits = 0;
    int tempNum = num;

    while (tempNum != 0)
    {
        tempNum /= 10;
        numberOfDigits++;
    }

    tempNum = num;
    while (tempNum != 0)
    {
        remainder = tempNum % 10;
        sum += pow(remainder, numberOfDigits);
        tempNum /= 10;
    }

    if (sum == originalNum)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int main()
{
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);
```

```
    if (isArmstrong(number))
    {
        printf("%d is an Armstrong number.\n", number);
    }
    else
    {
        printf("%d is not an Armstrong number.\n", number);
    }
    return 0;
}
```

Output:



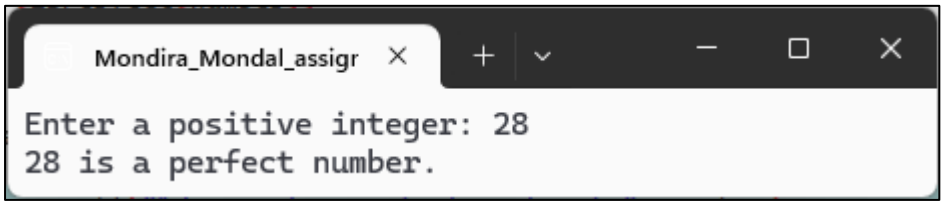
Assignment 3:

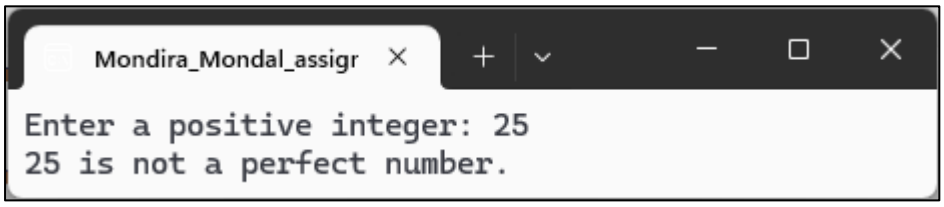
Write a C program that includes a user-defined function named isPerfect with the signature int **isPerfect(int num);**. A perfect number is a positive integer that is equal to the sum of its proper divisors, excluding itself. For example, 28 is a perfect number because the sum of its divisors (1, 2, 4, 7, 14) equals 28.

Source Code:

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

Output:





Assignment 4:

Write a C program that takes an integer input representing a month (1 to 12) and a year. Use a switch statement to display the number of days in that month, considering leap years.

Source Code:

```
#include <stdio.h>
int main()
{
    int month, year, numDays;
    printf("Enter month (1-12) and year: ");
    scanf("%d %d", &month, &year);
    if (month < 1 || month > 12)
    {
        printf("Invalid month entered. Please enter a month between 1 and
12.\n");
        return 1;
    }
    switch (month)
    {
    case 1:  // January
    case 3:  // March
    case 5:  // May
    case 7:  // July
    case 8:  // August
    case 10: // October
    case 12: // December
        numDays = 31;
        break;
    case 4:  // April
    case 6:  // June
    case 9:  // September
    case 11: // November
        numDays = 30;
        break;
    case 2: // February
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
            numDays = 29;
        else
            numDays = 28;
        break;
    }
    printf("Number of days in month %d of year %d: %d\n", month, year, numDays);
    return 0;
}
```

Output: