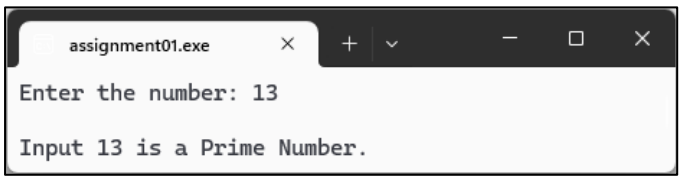
Assignment 1:

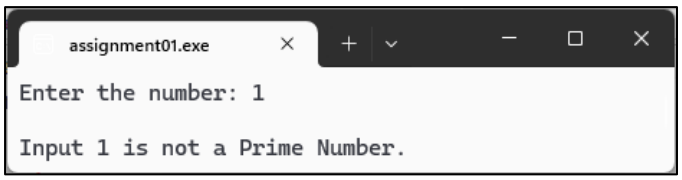
Write a C program that includes a user-defined function named isPrime with the signature **int isPrime(int num);** The function should take an integer as a parameter and return 1 if the number is prime and 0 otherwise.

Source Code:

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

int isPrime(int);

int main()
{
    int n;
    printf("\n\nEnter the number: ");
    scanf("%d", &n);
    if (isPrime(n))
    {
        printf("\n\nInput %d is a Prime Number.\n", n);
    }
    else
    {
        printf("\n\nInput %d is not a Prime Number.\n", n);
    }
    return 0;
}

int isPrime(int n)
{
    if (n <= 1)
        return 0;
    if (n == 2)
        return 1;
    if (n % 2 == 0)
        return 0;
    int temp = (int)sqrt(n);
    int i;
    for (i = 3; i <= temp; i += 2)
    {
        if (n % i == 0)
        {
            return 0;
        }
    }
    return 1;
}
```

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);
int count(int);

int main()
{
    int n;
    printf("Enter the number: ");
    scanf("%d", &n);

    if (isArmstrong(n))
    {
        printf("\nInput %d is a Armstrong Number.", n);
    }
    else
    {
        printf("\nInput %d is Not a Armstrong Number.", n);
    }

    return 0;
}

int count(int n)
{
    int count = 0;
    while (n > 0)
    {
        count++;
        n = n / 10;
    }
    return count;
}

int isArmstrong(int n)
{
    if (n < 0)
        return 0;
    if (n == 0)
        return 1;

    int power = count(n);
    int temp = n;
    int checker = 0;

    while (temp > 0)
    {
```

```
        int digit = temp % 10;
        checker = checker + (int)round(pow(digit, power));
        temp = temp / 10;
    }
    return n == checker;
}
```

Output:

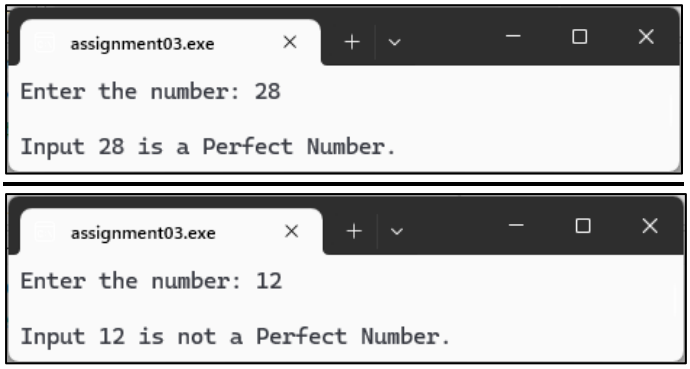
Assignment 3:

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);

int main()
{
    int n;
    printf("Enter the number: ");
    scanf("%d", &n);
    if (isPerfect(n))
    {
        printf("\nInput %d is a Perfect Number.", n);
    }
    else
    {
        printf("\n\nInput %d is not a Perfect Number.", n);
    }
    return 0;
}

int isPerfect(int n)
{
    if (n <= 1)
        return 0;
    int temp = 1;
    int i;
    for (i = 2; i <= n / 2; i++)
    {
        if (n % i == 0)
        {
            temp += i;
        }
    }
    return temp == n;
}
```

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, days;
    printf("Enter the month (1 to 12) and year: ");
    scanf("%d %d", &month, &year);
    switch (month)
    {
    case 1:  // jan
    case 3:  // mar
    case 5:  // may
    case 7:  // jul
    case 8:  // aug
    case 10: // oct
    case 12: // dec
        days = 31;
        break;
    case 4:  // apr
    case 6:  // jun
    case 9:  // sep
    case 11: // nov
        days = 30;
        break;
    case 2: // feb
        if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)))
        {
            days = 29;
        }
        else
        {
            days = 28;
        }
        break;
    default:
        printf("\nYou entered something wrong.");
        return 1;
    }
    printf("\nNumber of days: %d", days);
    return 0;
}
```

Output: