Print triangle-and allow user to set height of it in .Like in the following case its 4.

```
namespace Assignment1
{
 internal class Program
 {
  static void Main(string[] args)
  {
   //Console.WriteLine("Hello, World!");
   Console.WriteLine("Enter a number(height) and I'll print you a pyramid of the given height : ");
   int height = Convert.ToInt32 (Console.ReadLine());
   printPyramid(height);
   Console.ReadKey();
  }

  static void printPyramid(int height)
  {
   int increment = 1;
   for(int i =0;i < height;i++)
   {
    for(int j =0;j < height - i - 1; j++)
    {
     Console.Write(" ");
    }
    for(int k = i + increment; k > 0; k--)
    {
     Console.Write("*");
    }
    increment++;
    Console.WriteLine();
   }

  }
 }
}
```

Find valid date (MMDDYYYY) from string.

```
using Assignment2;
using Assignment2.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string input = "This is a sample text containing dates like 12312021 and 02282023.
Check them.";

        // Define a regular expression pattern for MMDDYYYY dates
        string pattern = @"\b(\d{2})(\d{2})(\d{4})\b";

        // Create a regex object
        Regex regex = new Regex(pattern);

        // Find all matches in the input string
        MatchCollection matches = regex.Matches(input);

        // Iterate through the matches
        foreach (Match match in matches)
        {
            int month = int.Parse(match.Groups[1].Value);
            int day = int.Parse(match.Groups[2].Value);
            int year = int.Parse(match.Groups[3].Value);

            // Check if the date is valid
            if (IsValidDate(month, day, year))
            {
                // Format the date as MMDDYYYY
                string formattedDate = $"{month:D2}{day:D2}{year:D4}";
                Console.WriteLine($"Valid date found: {formattedDate}");
            }
        }
    }

    // Function to check if a date is valid
    static bool IsValidDate(int month, int day, int year)
    {
        // Check if the year, month, and day form a valid date
        if (year >= 1000 && year <= 9999 && month >= 1 && month <= 12)
        {
            int[] daysInMonth = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

```
            // Check for leap year
            if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
            {
                daysInMonth[1] = 29; // February has 29 days in a leap year
            }

            if (day >= 1 && day <= daysInMonth[month - 1])
            {
                return true;
            }
        }

        return false;
    }
}
```