DOTNET ASSIGNMENT - Solution

1. **Print triangle - and allow user to set height of it in. Like in the following case it's 4**

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

Solution

```
using System;

class Pattern
{
    static void Main(string[] args)
    {
        Console.WriteLine("Enter the height of the triangle:");
        int height;
        while (!int.TryParse(Console.ReadLine(), out height) || height < 1)
        {
            Console.WriteLine("The height of the triangle must be greater than 0");
        }

        PrintTriangle(height);

        Console.ReadLine();
    }

    static void PrintTriangle(int height)
    {
        for (int i = 1; i <= height; i++)
        {
            for (int j = 0; j < height - i; j++)
            {
                Console.Write("  ");
            }

            for (int k = 0; k < 2 * i - 1; k++)
            {
                Console.Write(" * ");
            }

            Console.WriteLine();
        }
    }
}
```

Test Cases:

1. Test with height 1:
   - Input: 1
   - Output: *

2. Test with height 5:
   - Input: 4
   - Output:

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

3. Test with height 0:
   - Input: 0
   - Output: The height of the triangle must be greater than 0

4. Test with negative height:
   - Input: -5
   - Output: The height of the triangle must be greater than 0

2. Find valid date (MMDDYYYY) from string.

For example :-

Hdjsh asd2324234jghjsd hjsdg sdhk *12212021* idf32432 32423 d34234jh dfh

```
using System;
using System.Text.RegularExpressions;

class ValidString
{
    static void Main(string[] args)
    {
        Console.WriteLine("Enter a string:");
        string input = Console.ReadLine();

        string date = FindValidDate(input);

        if (date != null)
        {
            Console.WriteLine("Output: " + date);
        }
        else
```

```
        {
            Console.WriteLine("No valid date found in the string.");
        }

        Console.ReadLine();
    }

    static string FindValidDate(string input)
    {
        Regex regex = new Regex(@"\b(0[1-9]|1[0-2])(0[1-9]|[12]\d|3[01])(19|20)\d{2}\b");
        Match match = regex.Match(input);
        if (match.Success)
        {
            return match.Value;
        }
        return null;
    }
}
```

Test Cases:

1. Test with a valid date in the format MMDDYYYY:
    - Input: "Today's date is 03242024."
    - Expected Output: "03242024"
2. Test with multiple valid dates in the format MMDDYYYY:
    - Input: "The meeting is scheduled for 04152024 and the deadline is 05312024."
    - Expected Output: "04152024" (the first valid date found in the string)
3. Test with a valid date at the beginning of the string:
    - Input: "11292024 is Thanksgiving Day this year."
    - Expected Output: "11292024"
4. Test with no valid date in the string:
    - Input: "There are no valid dates in this sentence."
    - Expected Output: null