DOTNET ASSIGNMENT

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

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

Solution

```
using System;

class Question1
{
   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: 5
   - 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: -2
   - 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

Solution

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

class Question2
{
    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. String with a valid date:
   - Input: "The date is 12132020"
   - Output: 12132020

2. String with multiple dates:
   - Input: "The dates are 06262024 and 01102020"
   - Output: 06262024

3 String with no valid date
   - Input: "Here is the string with no valid date"
   - Output: No valid date found in the string.

4 String with invalid date format
   - Input: "The date is 2024/03/21"
   - Output: No valid date found in the string.