

# Assignment 1

Tianle Yuan

04/19/2023

## 01 Introduction to C# and Data Types

### Understanding Data Types

#### Test your Knowledge

1. What type would you choose for the following “numbers”?
  - A person’s telephone number **A: string**
  - A person’s height **A: float**
  - A person’s age **A: uint**
  - A person’s gender (Male, Female, Prefer Not To Answer) **A: enum**
  - A person’s salary **A: float**
  - A book’s ISBN **A: string**
  - A book’s price **A: float**
  - A book’s shipping weight **A: float**
  - A country’s population **A: long**
  - The number of stars in the universe **A: decimal or long**
  - The number of employees in each of the small or medium businesses in the United Kingdom (up to about 50,000 employees per business) **A: uint**

2. What are the difference between value type and reference type variables? What is boxing and unboxing?

**A:** Value type directly hold the value; reference type hold the memory address for this value.  
Value type can be int, float, bool, double, etc; reference type can be string, StringBuilder, object.  
Value type do not accept NULL type; reference type do.  
Value type includes struct and enum; reference type includes class, interface or array.  
Value type do not collected by garbage collector; reference type do.  
Value type is stored in stack memory; reference type is stored in heap memory.

**A:** Boxing is the process of converting a value type into a reference type. When a value type is boxed, the system creates a new object on the heap and wraps the value type instance within this object. The object is of the corresponding boxed type, which is a reference type that derives from the System.Object class. Boxing involves copying the value of the value type into the new object, leading to performance overhead.

Unboxing is the reverse process of boxing, where a reference type (boxed value) is converted back

into its original value type. During unboxing, the value is extracted from the boxed object and assigned to the value type variable. Unboxing requires an explicit cast and can lead to an `InvalidCastException` if the types are not compatible.

### 3. What is meant by the terms managed resource and unmanaged resource in .NET

**A:** Managed resources are those whose memory is managed by the .NET runtime, specifically the Garbage Collector (GC). The GC automatically handles the allocation, deallocation, and reclamation of memory for these resources, preventing memory leaks and ensuring efficient memory management.

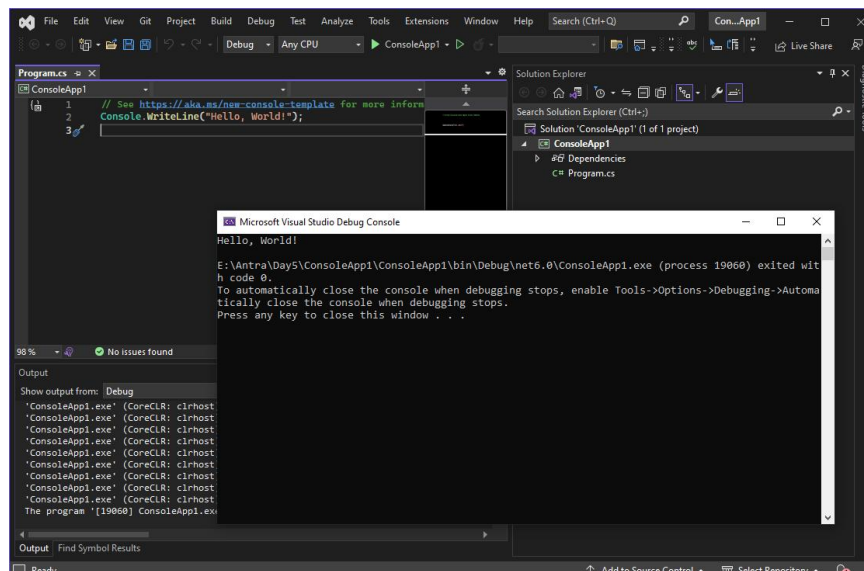
**Unmanaged Resources:** Unmanaged resources are those that are not managed by the .NET runtime or the Garbage Collector. They are typically resources that are allocated and deallocated directly by the operating system or external libraries, such as file handles, database connections, or graphics resources.

### 4. Whats the purpose of Garbage Collector in .NET?

**A:** Do not need to manually release memory; allocates objects on managed heap efficiently; and prevent memory leak.

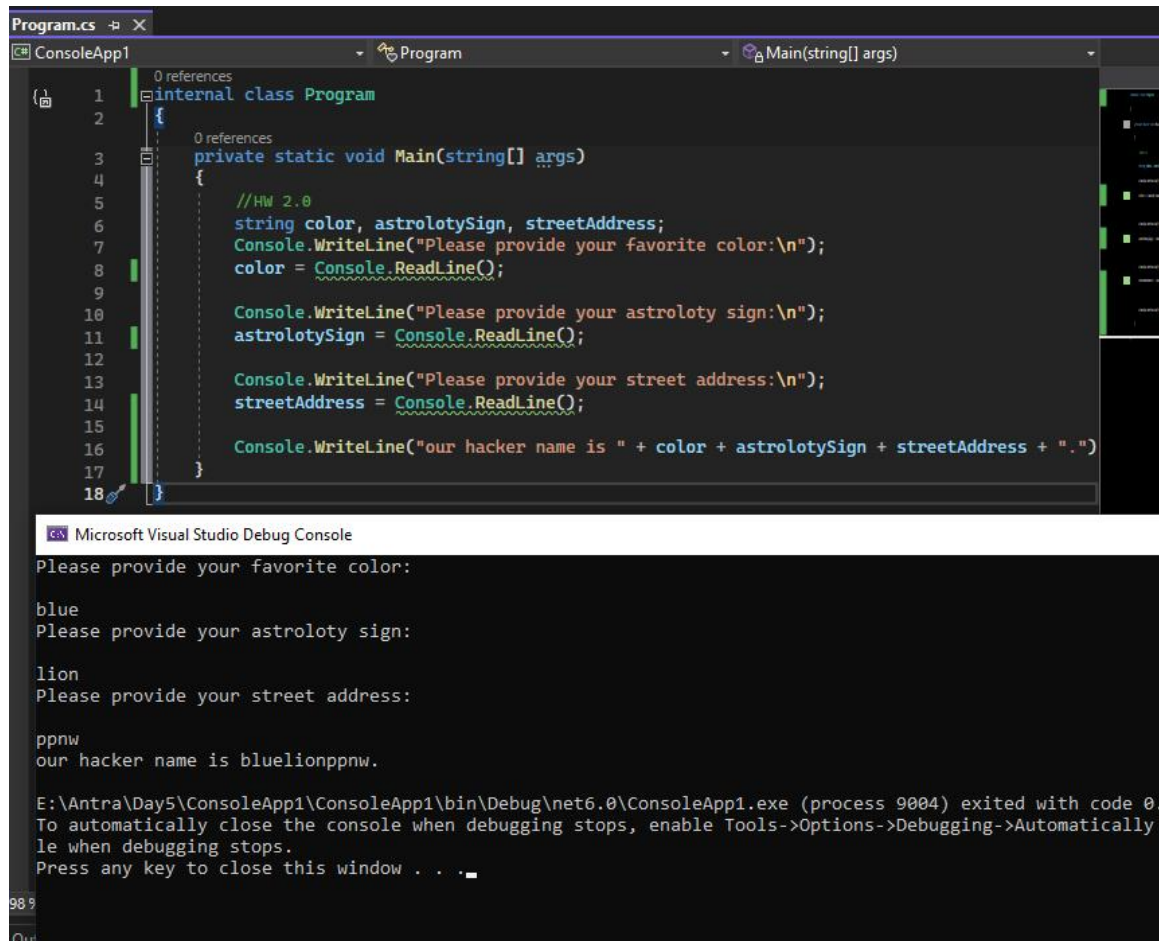
## Playing with Console App

**Modify your console application to display a different message. Go ahead and intentionally add some mistakes to your program, so you can see what kinds of error messages you get from the compiler. The more familiar you are with these messages, and what causes them, the better you'll be at diagnosing problems in your programs that you / didn't/ intend to add!**



**Using just the ReadLine and WriteLine methods and your current knowledge of variables, you can have the user pass in quite a few bits of information. Using this approach, create a console application that asks the user a few questions and then generates some custom**

output for them. For instance, your program could generate their "hacker name" by asking them their favorite color, their astrology sign, and their street address number. The result might be something like "Your hacker name is RedGemini480."



The screenshot shows a Visual Studio IDE with a C# console application. The code in `Program.cs` defines an internal class `Program` with a static `Main` method. The method prompts the user for their favorite color, astrology sign, and street address, then concatenates these into a "hacker name". The debug console shows the program's execution with user input: "blue", "lion", and "ppnw", resulting in the output "our hacker name is bluelionppnw."

```
Program.cs - X
ConsoleApp1
Program
Main(string[] args)

1 internal class Program
2 {
3     private static void Main(string[] args)
4     {
5         //HW 2.0
6         string color, astrologySign, streetAddress;
7         Console.WriteLine("Please provide your favorite color:\n");
8         color = Console.ReadLine();
9
10        Console.WriteLine("Please provide your astrology sign:\n");
11        astrologySign = Console.ReadLine();
12
13        Console.WriteLine("Please provide your street address:\n");
14        streetAddress = Console.ReadLine();
15
16        Console.WriteLine("our hacker name is " + color + astrologySign + streetAddress + ".")
17    }
18 }
```

Microsoft Visual Studio Debug Console

```
Please provide your favorite color:
blue
Please provide your astrology sign:
lion
Please provide your street address:
ppnw
our hacker name is bluelionppnw.

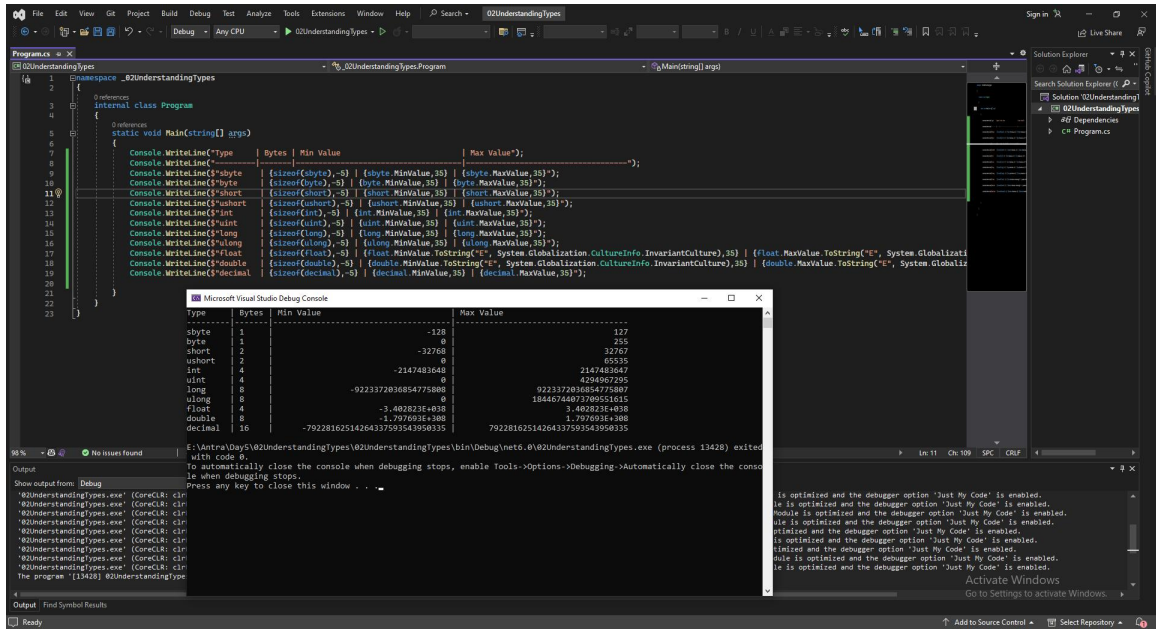
E:\Antra\Day5\ConsoleApp1\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe (process 9004) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically
le when debugging stops.
Press any key to close this window . . .
```

## Practice number sizes and ranges

1. Create a console application project named `/02UnderstandingTypes/` that outputs the number of bytes in memory that each of the following number types uses, and the minimum and maximum values they can have: `sbyte`, `byte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double`, and `decimal`.

A: Three points:

1. “{parameter, +(left align)/-(right align)}” for Composite formatting;
2. `sizeof(var)` for byte size;
3. For too large value, we use scientific expression with UTF-16 code units with specific culture: `double.MinValue.ToString("E", System.Globalization.CultureInfo.InvariantCulture)`



**2. Write program to enter an integer number of centuries and convert it to years, days, hours, minutes, seconds, milliseconds, microseconds, nanoseconds. Use an appropriate data type for every data conversion. Beware of overflows!**

**Input: 1**

**Output: 1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes**

**= 3155673600 seconds = 3155673600000 milliseconds = 3155673600000000**

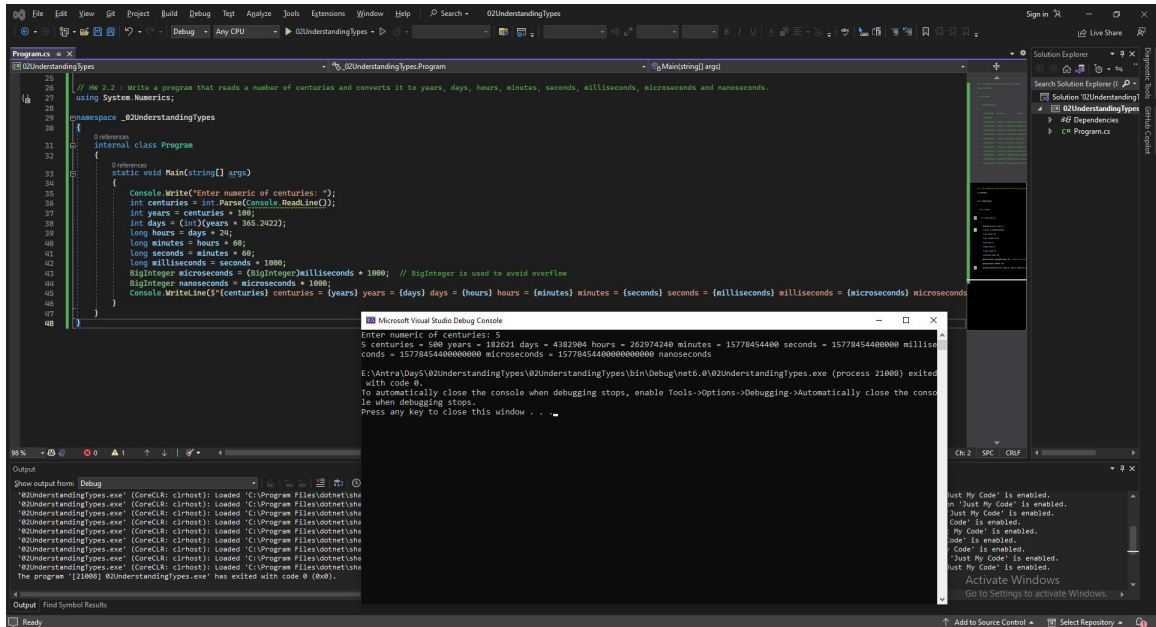
**microseconds = 3155673600000000000 nanoseconds**

**Input: 5**

**Output: 5 centuries = 500 years = 182621 days = 4382904 hours = 262974240**

**minutes = 15778454400 seconds = 15778454400000 milliseconds = 15778454400000000**

**microseconds = 15778454400000000000 nanoseconds**



## Explore following topics

- C# Keywords
- Main() and command-line arguments
- Types (C# Programming Guide)
- Statements, Expressions, and Operators
- Strings (C# Programming Guide)
- Nullable Types (C# Programming Guide)
- Nullable reference types

## Controlling Flow and Converting Types

### Test your Knowledge

#### 1. What happens when you divide an int variable by 0?

**A:** error of “System.DivideByZeroException” for undefined dividing.

#### 2. What happens when you divide a double variable by 0?

**A:** either positive infinity, negative infinity, or NaN (not-a-number), depending on the sign of the dividend.

#### 3. What happens when you overflow an int variable, that is, set it to a value beyond its range?

**A:** if the int variable exceeds its maximum value (2,147,483,647), it wraps around to its minimum value (-2,147,483,648) and continues from there. The overflow is like a circle.

**4. What is the difference between `x = y++`; and `x = ++y`;**

**A:** The first one is a post-increment operation. The value of y is assigned to x, before y is incremented by 1. The second one is a pre-increment operation. The value of y is incremented by 1 before it is assigned to x.

**5. What is the difference between `break`, `continue`, and `return` when used inside a loop statement?**

**A:** `break` is used to exit the loop immediately; `continue` is used to skip the current iteration and proceed to the next one; `return` is used to exit the method containing the loop and return a value (if specified) to the caller.

**6. What are the three parts of a `for` statement and which of them are required?**

**A:** Three parts: the initializer, the condition, and the iterator. All of them are optional.

**7. What is the difference between the `=` and `==` operators?**

**A:** The `=` operator is the assignment operator, which assigns a value to a variable. The `==` operator is the equality operator, which checks if two values are equal.

**8. Does the following statement compile? `for ( ; true; ) ;`**

**A:** It is an infinite loop with no loop body.

**9. What does the underscore `_` represent in a `switch` expression?**

**A:** In a `switch` expression, the underscore `_` represents a discard pattern. It is used to catch any value that hasn't matched any previous pattern in the `switch` expression. (like default in C# `Switch Statement`)

**10. What interface must an object implement to be enumerated over by using the `foreach` statement?**

**A:** The object must implement the `IEnumerable` interface (or `IEnumerable<T>` for a specific type `T`), to be enumerated over by using the `foreach` statement.

## Practice loops and operators

**1. FizzBuzz** is a group word game for children to teach them about division. Players take turns to count incrementally, replacing any number divisible by three with the word `/fizz/`, any number divisible by five with the word `/buzz/`, and any number divisible by both with `/fizzbuzz/`.

Create a console application in Chapter03 named `Exercise03` that outputs a simulated FizzBuzz game counting up to 100. The output should look something like the following screenshot:

What will happen if this code executes?

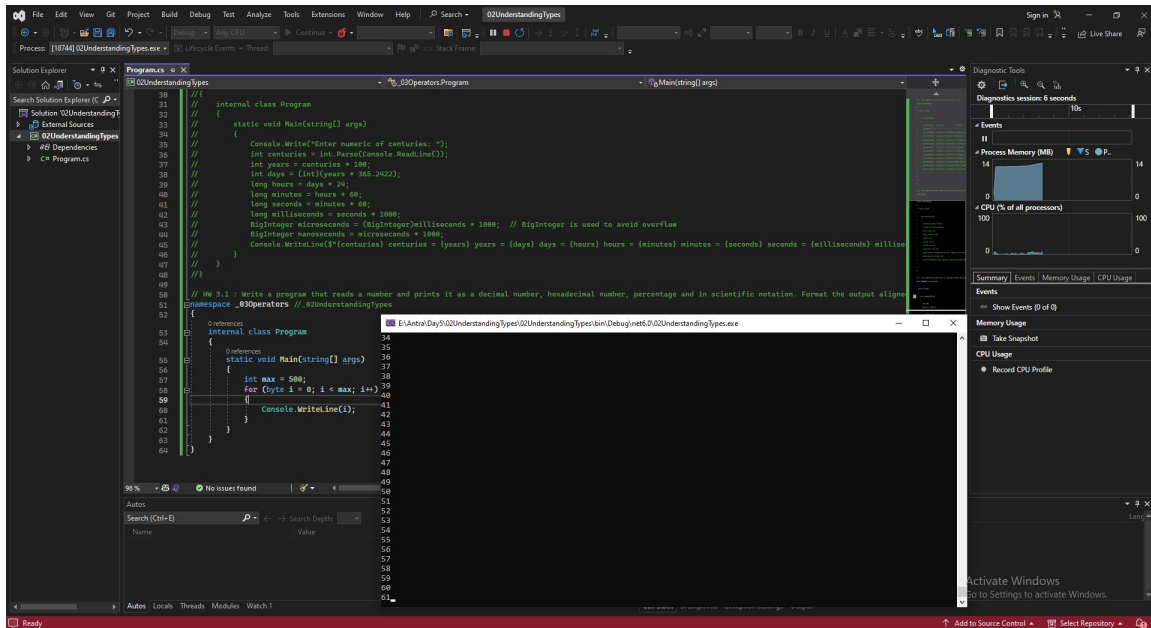
```
int max = 500;
for (byte i = 0; i < max; i++)
{
    WriteLine(i);
}
```

Create a console application and enter the preceding code. Run the console application



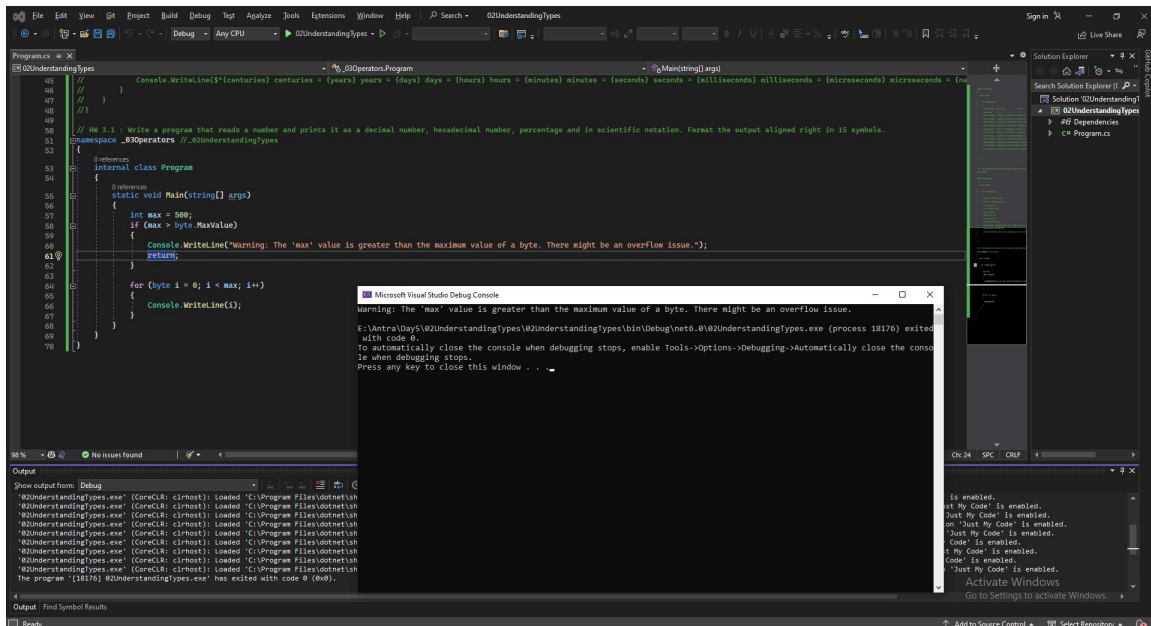
and view the output. What happens?

A: We should see an endless loop because of byte type overflow. The picture can be see below.



What code could you add (don't change any of the preceding code) to warn us about the problem?

A: We can just add an if statement to check the overflow before the loop:



Your program can create a random number between 1 and 3 with the following code:

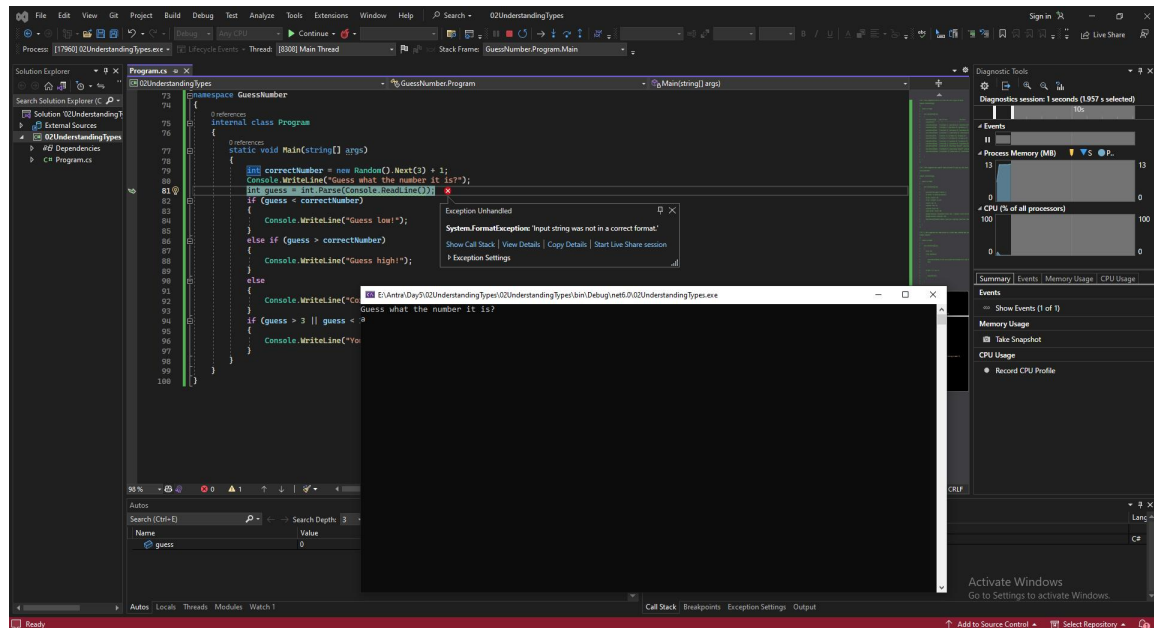
```
int correctNumber = new Random().Next(3) + 1;
```

Write a program that generates a random number between 1 and 3 and asks the user to guess what the number is. Tell the user if they guess low, high, or get the correct answer.

Also, tell the user if their answer is outside of the range of numbers that are valid guesses (less than 1 or more than 3). You can convert the user's typed answer from a string to an int using this code:

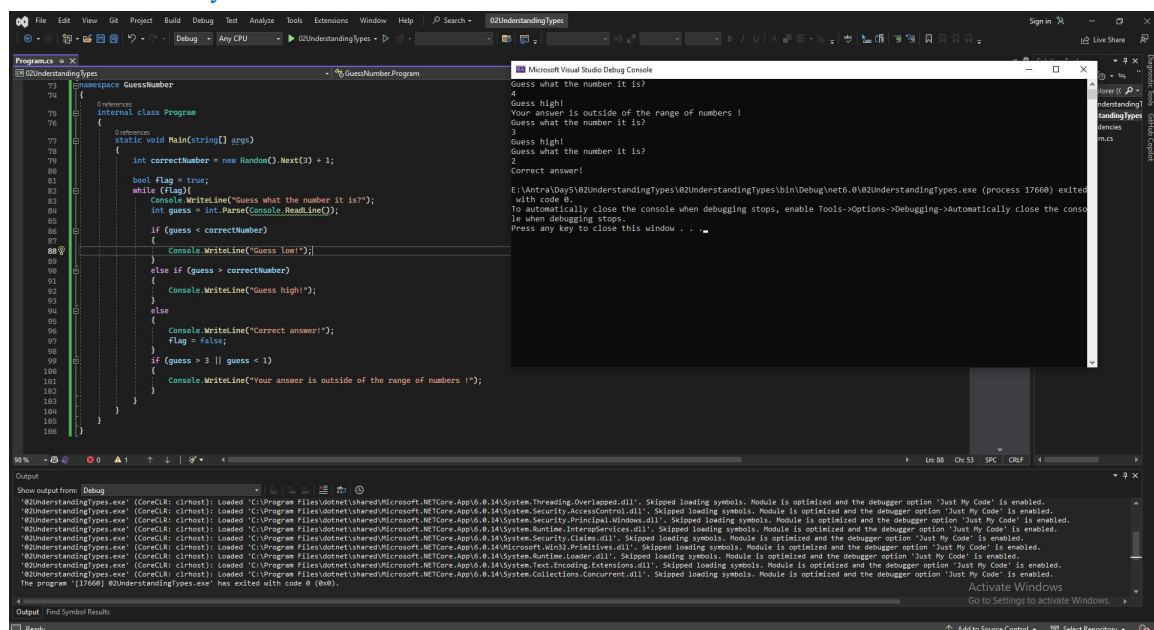
```
int guessedNumber = int.Parse(Console.ReadLine());
```

Note that the above code will crash the program if the user doesn't type an integer value.



For this exercise, assume the user will only enter valid guesses.

A: Here is my final code:



2. Print-a-Pyramid. Like the star pattern examples that we saw earlier, create a program that will print the following pattern: If you find yourself getting stuck, try recreating the two examples that we just talked about in this chapter first. They're simpler, and you can

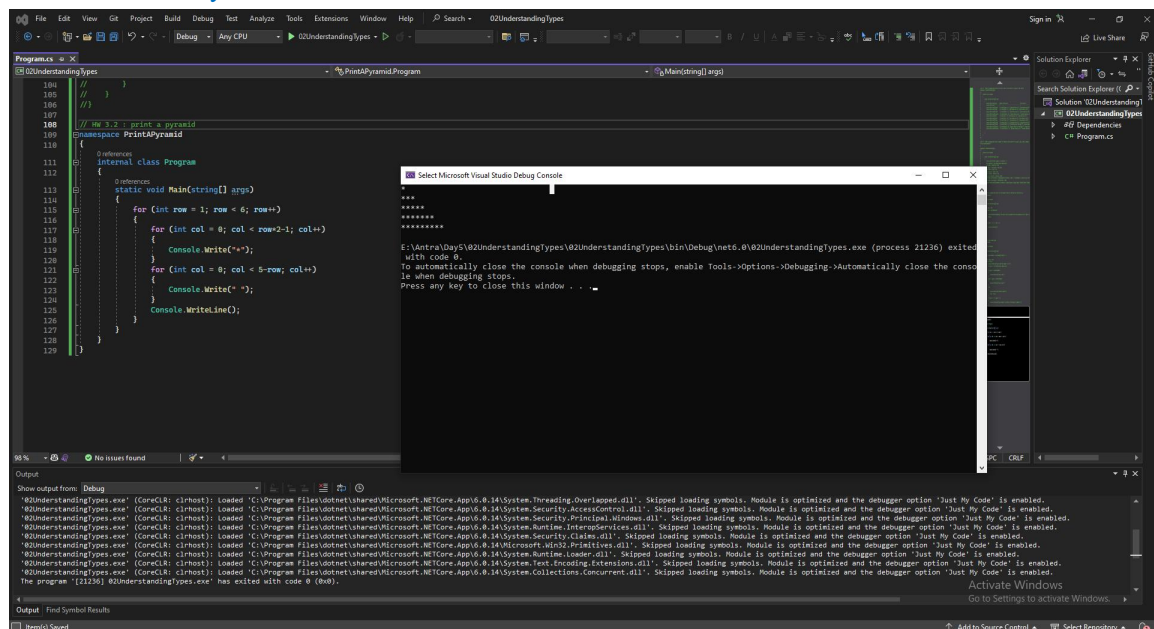


compare your results with the code included above.

This can actually be a pretty challenging problem, so here is a hint to get you going. I used three total loops. One big one contains two smaller loops. The bigger loop goes from line to line. The first of the two inner loops prints the correct number of spaces, while the second inner loop prints out the correct number of stars.

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

A: Here is my final code:



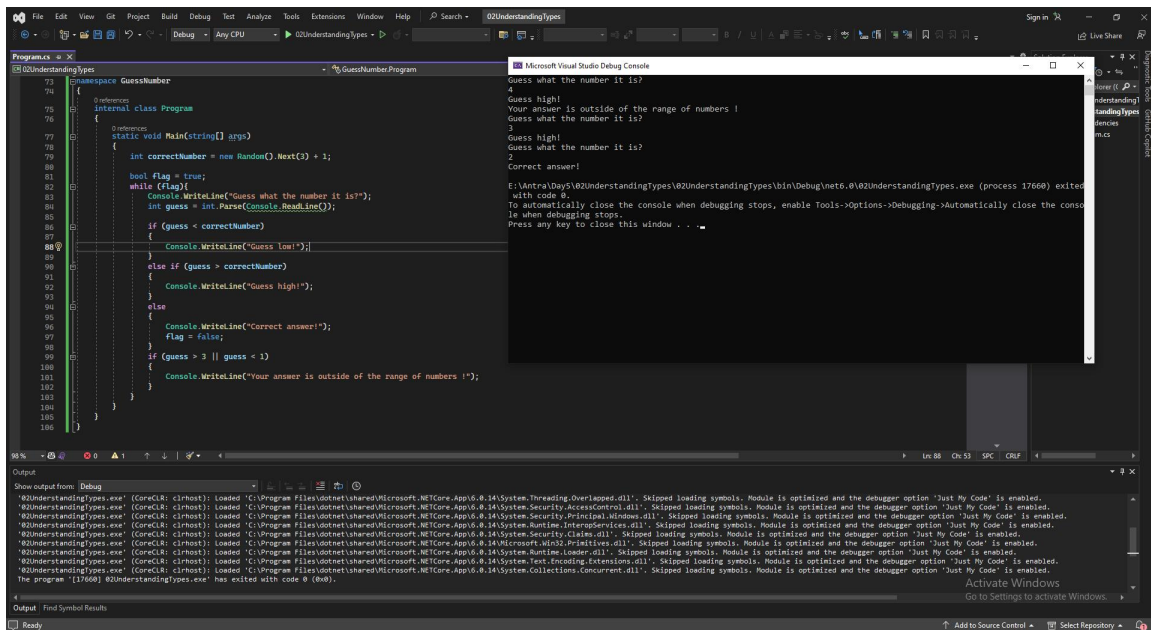
3. Write a program that generates a random number between 1 and 3 and asks the user to guess what the number is. Tell the user if they guess low, high, or get the correct answer. Also, tell the user if their answer is outside of the range of numbers that are valid guesses (less than 1 or more than 3). You can convert the user's typed answer from a string to an int using this code:

```
int guessedNumber = int.Parse(Console.ReadLine());
```

Note that the above code will crash the program if the user doesn't type an integer value.

For this exercise, assume the user will only enter valid guesses.

A: This question is the same one as we finished above in question 3.1.2. Here is my final code:

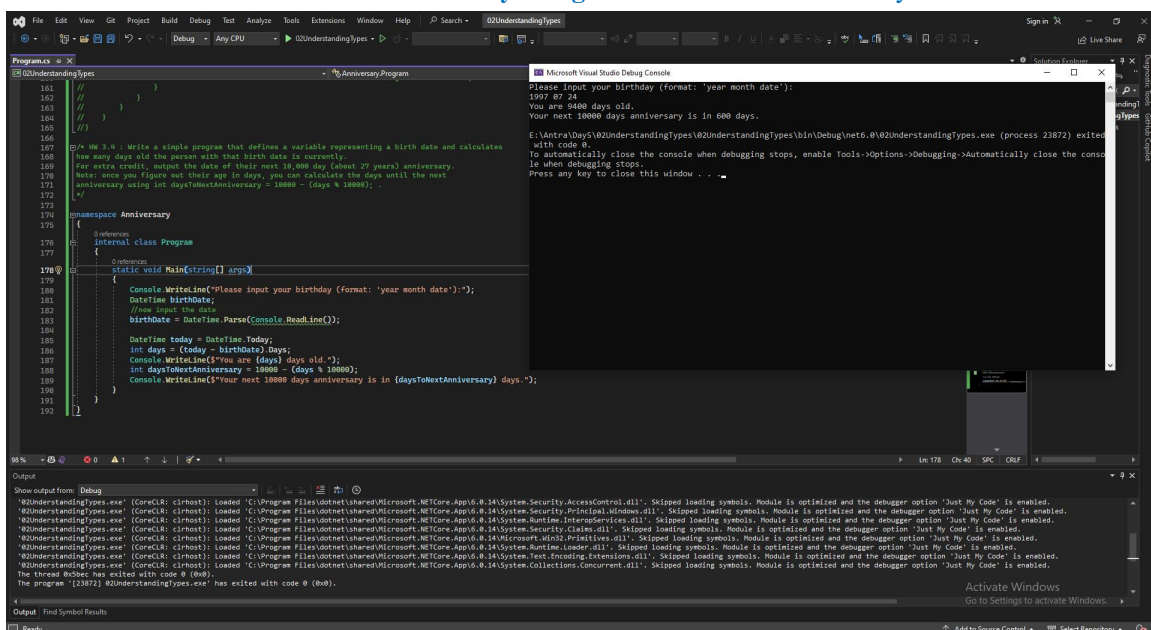


4. Write a simple program that defines a variable representing a birth date and calculates how many days old the person with that birth date is currently.

For extra credit, output the date of their next 10,000 day (about 27 years) anniversary.

Note: once you figure out their age in days, you can calculate the days until the next anniversary using  $\text{int daysToNextAnniversary} = 10000 - (\text{days} \% 10000)$ ; .

**A: We need to define the birthDate by using DateTime class. Here is my final code:**

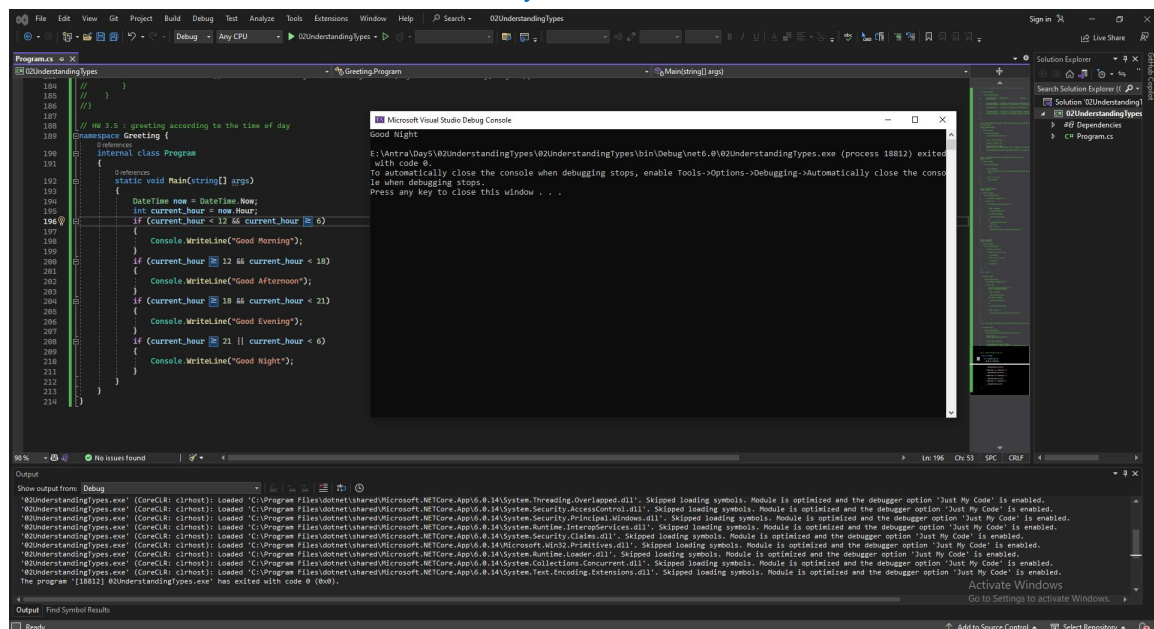


5. Write a program that greets the user using the appropriate greeting for the time of day.

Use only if , not else or switch , statements to do so. Be sure to include the following greetings:

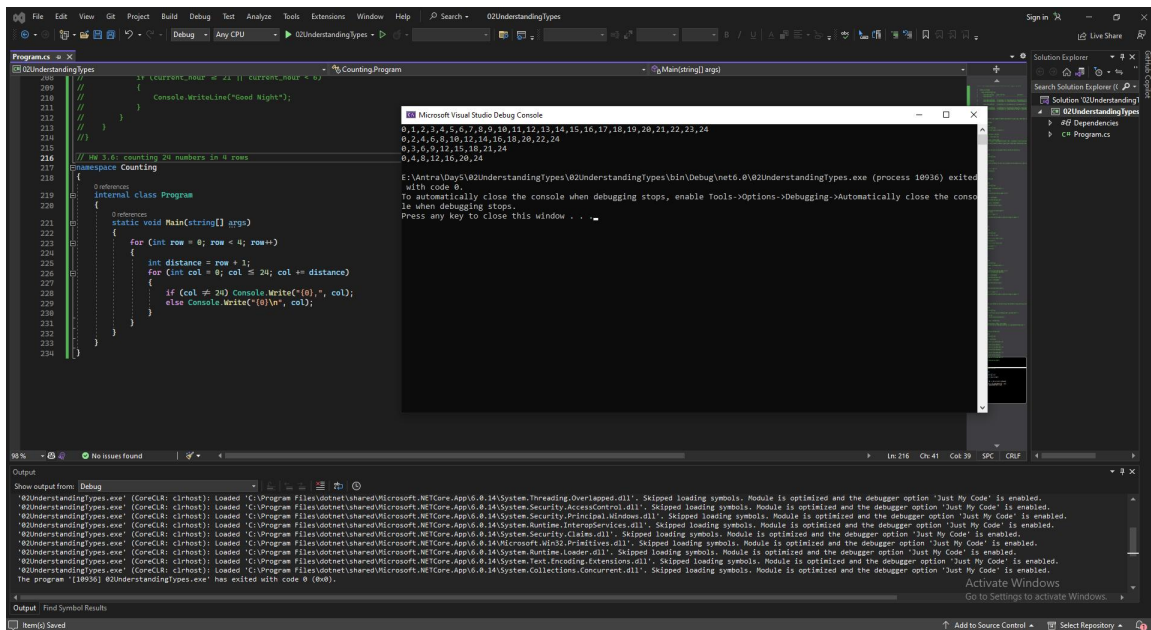
## "Good Night"

**A: Current time is 12:33 am. Here is my final code:**



**0,4,8,12,16,20,24**

**A:** Here is my final code:



## Explore following topics

- **C# operators**
- **Bitwise and shift operators**
- **Statement keywords**
- **Casting and type conversions**
- **Fundamentals of garbage collection**
- **\$ - string interpolation**
- **Formatting types in .NET**
- **Iteration statements**
- **Selection statements**