Review

R100 – Binary, Hex & RPN

- We will do these short low-stakes tests in the final lesson of each week
- They are there for you to see how well you have learnt and understood recent content
- **You must write your answers on paper**
- Once marked:
  1. Report your score using the Google Form on Classroom
  2. Write your own reflections and targets in the generated tracker Google Doc

Starter

```
static void Main(string[] args)
{
    int cakeEaten = "";
    cakeEaten = Console.ReadKey();
    Console.WriteLine("Did you eat my cake?");
    switch (cakEaten)
    {
        case "Yes"
            Console.Writeline("That's naughty!");
            break;
        case "No":
            Console.WriteLine(You liar");
            Console.ReadKey();
}
```

In pairs, can you identify all 9 mistakes in this code?

Programming catch-up

You have 20 minutes to catch up on any of the programming worksheets so far.

- W100 – C# Variables
- W101 – C# If
- W102 – C# For

Ask for extension activities if you have finished.

C# For

Compound assignment

```
int i = 0;
```

- What code would you write to add `1` to `i`?

Compound assignment

```
int i = 0;
```

- What code would you write to add 1 to i?
  - `i = i + 1;`

Compound assignment

```
int i = 0;
```

- What code would you write to add `1` to `i`?
  - `i = i + 1;`
- We can combine operators and assignment; this is known as compound assignment
  - `i += 1;`
  - `i -= 1;`
  - `i *= 10;`
  - `i /= 5;`

Increment/decrement operators

- If we only want to increment or decrement by 1, we can write it even more concisely
  - `i++;`
  - `i--;`
- `i++` does the same thing has `i += 1` which does the same thing as `i = i + 1`

Motivation

What code would you write to produce the following output?

```
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
```

Motivation

What code would you write to produce the following output?

```
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
```

```
static void Main(string[] args)
{
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
}
```

Motivation

What code would you write to produce the following output?

```
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
I love Computer Science
```

```
static void Main(string[] args)
{
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
    Console.WriteLine("I love Computer Science");
}
```

Perhaps there's a better way...

`for` loops

```
static void Main(string[] args)
{
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine("I love Computer Science");
    }
}
```

- `i` is a **counter variable**. It can have any identifier, but `i` is typically used by convention
- This is an example of **definite iteration** (or **fixed iteration**), because the number of iterations is known before the loop starts

Structure of a `for` loop

```
for (int i = 0; i < 10; i++)
```

There are three elements:

Structure of a `for` loop

```
for (int i = 0; i < 10; i++)
```

There are three elements:

- Initialiser
  - `int i = 0`
  - Executed once before the loop starts, and is used to declare and initialise the counter variable

Structure of a `for` loop

```
for (int i = 0; i < 10; i++)
```

There are three elements:

- Initialiser
  - `int i = 0`
  - Executed once before the loop starts, and is used to declare and initialise the counter variable
- Condition
  - `i < 10`
  - Evaluated after each loop; the loop continues if it evaluates to `true`

Structure of a `for` loop

```
for (int i = 0; i < 10; i++)
```

There are three elements:

- Initialiser
  - `int i = 0`
  - Executed once before the loop starts, and is used to declare and initialise the counter variable
- Condition
  - `i < 10`
  - Evaluated after each loop; the loop continues if it evaluates to `true`
- Iterator
  - `i++`
  - Executed at the end of each iteration, typically used to increment the counter (`i`)

Exercise

Pick a challenge to attempt. **You have 5 minutes.**

1. Write a program that displays `"I must always hand in my homework on time"` 100 times
2. Write a program that displays any message requested by the user 100 times
3. Write a program that displays any message requested by the user, with the number of displays it is displayed also specified by the user

Worksheet

W103 – C# For