SA 2

- Week beginning 17th November (5 weeks away)
- Programming only
- Topics include everything up to **L120 – Lists 2**
  - See BPCompSci for a full list of topics we've covered; read upwards
- If you haven't already, you need to start revising now

Built-in Functions for String Manipulation

Strings

- A `string` is a sequence of `char`s

```
static void Main(string[] args)
{
  string name = "Imani";
  char c = name[2];

  Console.WriteLine(c);
}
```

This program outputs a

Strings

- A `string` is a sequence of `char`s

```
static void Main(string[] args)
{
  string name = "Imani";
  char c = name[0];

  Console.WriteLine(c);
}
```

What does this program output?

Strings

- A `string` is a sequence of `char`s

```
static void Main(string[] args)
{
  string name = "Imani";
  char c = name[0];

  Console.WriteLine(c);
}
```

What does this program output? `I`

Strings

- A `string` is a sequence of `char`s

```
static void Main(string[] args)
{
  string name = "Imani";
  char c = name[0];

  Console.WriteLine(c.ToString() + name[3].ToString());
}
```

What does this program output?

Strings

- A `string` is a sequence of `char`s

```
static void Main(string[] args)
{
  string name = "Imani";
  char c = name[0];

  Console.WriteLine(c.ToString() + name[3].ToString());
}
```

What does this program output? In

Subroutines that return values

- So far, we have covered **procedures** with **parameters** (inputs)
- The only form of ‘output’ we have is writing to the console

Subroutines that return values

- So far, we have covered **procedures** with **parameters** (inputs)
- The only form of ‘output’ we have is writing to the console
- Today, we will use built-in functions that actually return a value in the code
- A **function** is a subroutine that **returns** a value

`ToUpper` and `ToLower`

- `ToUpper` converts all letters in a string to upper case
- `ToLower` converts all letters in a string to lower case
- These are both subroutines (specifically functions)

`ToUpper` and `ToLower`

- `ToUpper` converts all letters in a string to upper case
- `ToLower` converts all letters in a string to lower case
- These are both subroutines (specifically functions)

```
Console.Write("Enter word: ");
string word = Console.ReadLine();
Console.WriteLine("In upper case: " + word.ToUpper());
Console.WriteLine("In lower case: " + word.ToLower());
```

`Length`

- `Length` is the number of characters in a string
- It is not a subroutine, instead it is a **property**
  - We will see these in more depth next term
- As it is not a subroutine, we **do not** put brackets at the end of it

Length

- `Length` is the number of characters in a string
- It is not a subroutine, instead it is a **property**
  - We will see these in more depth next term
- As it is not a subroutine, we **do not** put brackets at the end of it

```
Console.Write("Enter word: ");
string word = Console.ReadLine();
int length = word.Length;
Console.WriteLine("There are " + length + " characters");
```

Exercise

Write a program that:

- Asks the user for a word and stores it in a local variable
- Lets them between the following options:
  - Displaying the length of the word
  - Displaying the word in upper case
  - Displaying the word in lower case

You have **10 minutes**

- `word.Length`
- `word.ToUpper()`
- `word.ToLower()`

Extension: Investigate the `String.Replace()` function and use it to count the number of non-space characters in the phrase

Substring

- Returns a **substring** of a string, specified by the starting position (counting from 0) and, optionally, the desired length

```
.Substring(startIndex)
.Substring(startIndex, length)
```

- When the `length` parameter is omitted, the substring contains all of the string from `startIndex` onwards

`word.Substring(0, 2)`

Input: `"Hello World"`

Returns:

Substring

- Returns a **substring** of a string, specified by the starting position (counting from 0) and, optionally, the desired length

```
.Substring(startIndex)
.Substring(startIndex, length)
```

- When the `length` parameter is omitted, the substring contains all of the string from `startIndex` onwards

`word.Substring(0, 2)`

Input: `"Hello World"`

Returns: He

Substring

- Returns a **substring** of a string, specified by the starting position (counting from 0) and, optionally, the desired length

```
.Substring(startIndex)
.Substring(startIndex, length)
```

- When the `length` parameter is omitted, the substring contains all of the string from `startIndex` onwards

`word.Substring(0, 2)`

Input: `"Hello World"`

Returns: He

`word.Substring(2, 4)`

Input: `"Hello World"`

Returns:

Substring

- Returns a **substring** of a string, specified by the starting position (counting from 0) and, optionally, the desired length

```
.Substring(startIndex)
.Substring(startIndex, length)
```

- When the `length` parameter is omitted, the substring contains all of the string from `startIndex` onwards

`word.Substring(0, 2)`

Input: `"Hello World"`

Returns: `He`

`word.Substring(2, 4)`

Input: `"Hello World"`

Returns: `"llo "`

`Substring` Activity

Input string	Function call	Return value
`"Wingardium Leviosa"`	`word.Substring(0, 1)`	
`"Wingardium Leviosa"`	`word.Substring(1, 1)`	
`"Wingardium Leviosa"`	`word.Substring(9, 3)`	
`"Wingardium Leviosa"`	`word.Substring(11, 2)`	
`"Wingardium Leviosa"`	`word.Substring(3, 3)`	
`"Wingardium Leviosa"`	`word.Substring(6, 4)`	

`Substring` Activity

Input string	Function call	Return value
`"Wingardium Leviosa"`	`word.Substring(0, 1)`	`"W"`
`"Wingardium Leviosa"`	`word.Substring(1, 1)`	`"i"`
`"Wingardium Leviosa"`	`word.Substring(9, 3)`	`"m L"`
`"Wingardium Leviosa"`	`word.Substring(11, 2)`	`"Le"`
`"Wingardium Leviosa"`	`word.Substring(3, 3)`	`"gar"`
`"Wingardium Leviosa"`	`word.Substring(6, 4)`	`"dium"`

IndexOf

- Returns the starting position in the string of the first occurrence of the specified string

```
Console.WriteLine("We're looking for 'ham'");
Console.Write("Enter phrase: ");
string word = Console.ReadLine();
int index = word.IndexOf("ham");
Console.WriteLine("Found at " + index);
```

What is the index given the following input?

1. `"ham"`
2. `"hampton court"`
3. `"Birmingham"`
4. `"Ham"`

IndexOf

- Returns the starting position in the string of the first occurrence of the specified string

```
Console.WriteLine("We're looking for 'ham'");
Console.Write("Enter phrase: ");
string word = Console.ReadLine();
int index = word.IndexOf("ham");
Console.WriteLine("Found at " + index);
```

What is the index given the following input?

1. `"ham"` `0`
2. `"hampton court"` `0`
3. `"Birmingham"` `7`
4. `"Ham"` `-1`

Password generator

Automatically generate a password from information provided by the user.

Format	Example
1st letter of first name in upper case	**H**arry
2nd letter of surname in upper case	P**o**tter
Last 2 digits of year of both	19**80**
2nd and 3rd letters of favourite colour	r**ed**
1st 3 letters letters of street name	**Pri**vet Drive
1st digit of their shoe size	**9**

Password: HO80edPri9

Extension: Extract random characters of the personal information instead

String.Compare

- Returns an integer depending on which string comes before the other alphabetically
  - Greater than 0 if first is after the second
  - 0 if they are the same
  - Less than 0 if the first is before the second

```
string string1 = "Abacus";
string string2 = "Able";
if (String.Compare(string1, string2) < 0)
{
  Console.WriteLine(string1 + " is before " + string2);
}
```

Casting between `char` and `int`

```
char userChar = 'A';
int userNum = 103;

Console.WriteLine((int) userChar);
Console.WriteLine((char) userNum);
```

Worksheet

W107 - C# Built-in Functions

Text-based adventure game

Introduction

- Mini project for this half term, using the skills you've learned so far
  - Practical skills
    - Selection
    - Iteration
    - Procedures
    - Random numbers
  - Employability/transferable skills
    - Creativity
    - Planning
    - Problem solving
    - Independence
    - Resilience

Requirements

- Must be a C#.NET Console Application (so lots of beautiful ASCII art please!)
- Create a basic adventure game based on choices the user makes:
  - There must be a conclusion for each path through the game
  - There must be at least one random element in the game
  - The story of the game is your choice, but the user must be asked to make a choice on at least 3 occasions, each with at least two options that lead to different paths
  - There must be at least one ‘successful’ conclusion

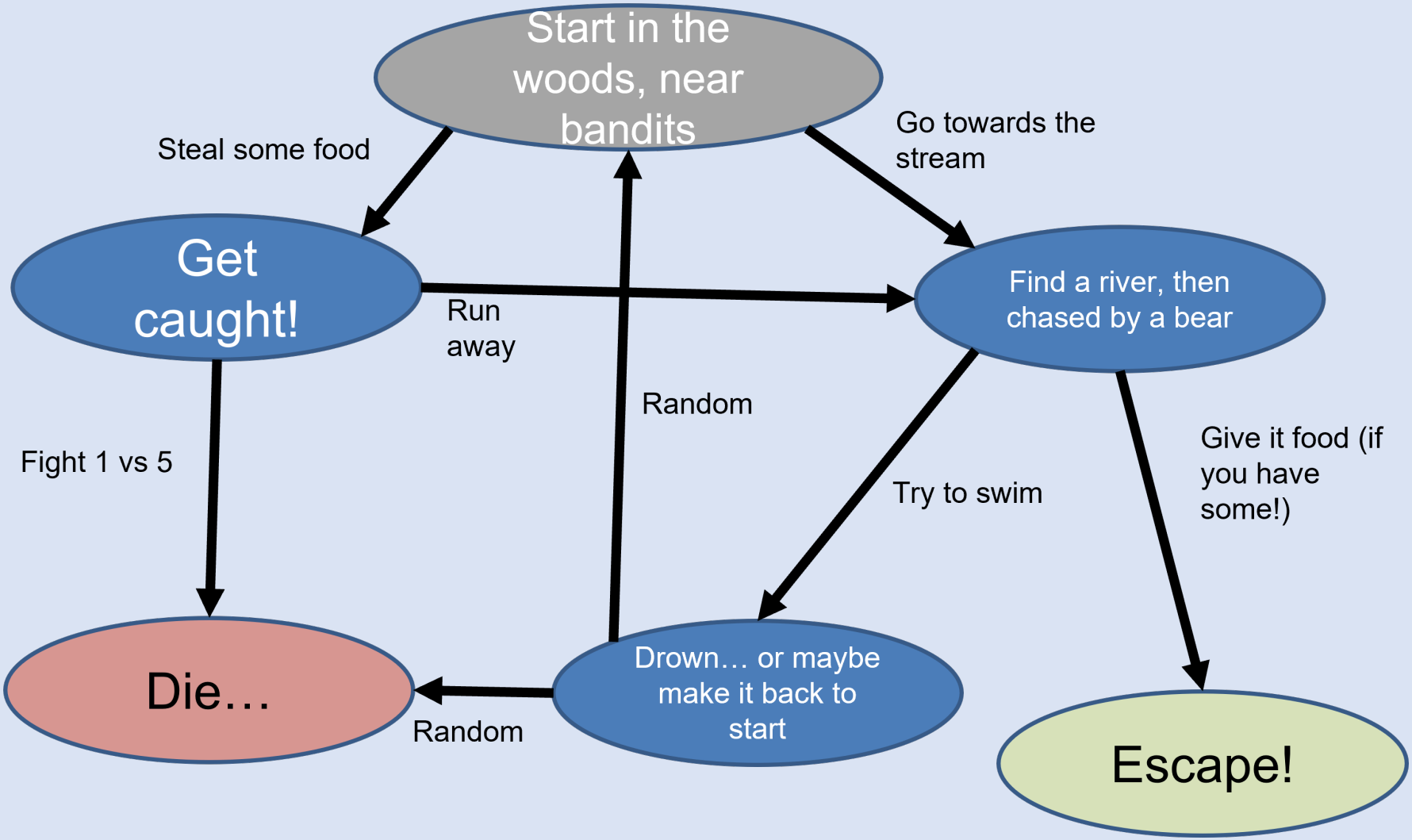
Advice – make a menu

```
static void Main(string[] args)
{
    Console.WriteLine(“Lost in the woods… Nearby there is a Bandit camp. What will you do?");
    Console.WriteLine("[C] Go towards the camp");
    Console.WriteLine("[W] Head further into the woods");
    Console.WriteLine("[R] Follow the sound of running water, towards the river");
    Console.WriteLine("[E] Exit game");
    string userChoice = Console.ReadLine().ToUpper();
    while (userChoice != "E")
    {
        if (userChoice == "C")
        {
            Camp();
            userChoice = "E";
        }
        else if (userChoice == "W")
        {
            Woods();
            userChoice = "E";
        }
        else if (userChoice == "R")
        {
            River();
            userChoice = "E";
        }
        else
        {
            Console.WriteLine("Not a valid choice, try again");
        }
    }
    Console.WriteLine("Thank you for playing");
    Console.ReadKey();
}
```

Advice – make a plan

- Come with a theme idea – it can be as simple or as complex as you like
- Previous ideas include:
  - Breaking into a bank vault
  - Escaping the wilderness whilst being chased by bears
  - Trying to cook spaghetti
  - Surviving a haunted house
  - Computer Science homework simulator

Plan it out


Start in the woods, near bandits
Steal some food
Go towards the stream
Get caught!
Run away
Find a river, then chased by a bear
Random
Fight 1 vs 5
Give it food (if you have some!)
Try to swim
Die…
Drown… or maybe make it back to start
Random
Escape!

Programming

- User inputs – use different variable types
- `if` statements, `switch`/`case`
- Random number generation
- Subroutines
- Loops `for`/`while`/`do`

Usability

- Make sure it is easy for someone else to play
  - Are the instructions clear?
  - Is it clear what the user should input?
  - Is the story clear?
  - Can it be played in 2 minutes?

Deadline

Tuesday 4th November 2025