Switch Statements

Using multiple `else if` statements can get unwieldy pretty quickly. Imagine writing an `else if` statement for every possible number of guests. And you invited 20 people. You have to write a lot of repetitive code, which is hard to read and prone to errors.

If it's necessary to evaluate several conditions with their own unique output, a *switch statement* is the way to go. Switch statements allow for compact control flow structures by evaluating a single expression and executing code blocks based on a matched case.

In C#, we write a switch statement using the following syntax:

```
string color;

switch (color)
{
   case "blue":
      // execute if the value of color is "blue"
      Console.WriteLine("color is blue");
      break;
   case "red":
      // execute if the value of color is "red"
      Console.WriteLine("color is red");
      break;
   case "green":
      // execute if the value of color is "green"
      Console.WriteLine("color is green");
      break;
   default:
      // execute if none of the above conditions are met
      break;
}
```

In this example, the program checks to see what the value of `color` equals. If it matches any of the specified cases, it will execute the code associated with that case. In C#, the *break*keyword allows programs to exit a block when a specific condition is met. If none of the conditions are met, the code inside the `default` case will run.

When using a switch statement, make sure to pay attention to:

> Cases: rather than writing out each condition, if we're evaluating one value we use *cases* to specify different potential values.
>
> Braces: rather than each case having its own code block, the entire statement lives within one set of braces `{}`.
>
> Colons: to distinguish between different cases, we state the case value, followed by a colon `:`. The code that should execute if that case is met follows.
>
> Break: Each case code needs to end with a `break` keyword.
>
> Default: Every switch statement needs a `default` case.

☑**Instructions**

1.
You want to build a simple movie recommender that gives the top movie in a particular genre.

First, create a `string` variable named `genre` and save the value `"Horror"` to it.

Hint

To create a variable with a string value, define the data type as `string`:

```
string language = "French";
```

2.
Create a switch statement using `genre`. Don't add any cases to the code block yet.

Hint

To write a switch statement:

```
switch (color)
{
// rest of it will go in here
}
```

3.

Next, add the following movie genres as cases to the switch statement. Make sure to also add a `default` case. Add a `break` statement to each case.

Genres:

Drama

Comedy

Adventure

Horror

Science Fiction

Hint

In C#, we write a switch statement using the following syntax:

```
string color;

switch (color)
{
   case "blue":
      // execute if the value of color is "blue"
      break;
   case "red":
      // execute if the value of color is "red"
      break;
   case "green":
      // execute if the value of color is "green"
      break;
   default:
      // execute if none of the above conditions are met
      break;
}
```

4.
Next, add `Console.WriteLine()` statements to each case in the switch statement so that the program prints out different movie titles based on the selected genre. For the `default` case, print “No movie found”.

Take a look at the following table to see the top movie for five different genres:

Genre	Movie
Drama	Citizen Kane
Comedy	Duck Soup
Adventure	King Kong
Horror	Psycho
Science Fiction	2001: A Space Odyssey

Hint

In C#, we write a switch statement using the following syntax:

```
string color;

switch (color)
{
   case "blue":
      Console.WriteLine("color is blue");
      break;
   case "red":
      Console.WriteLine("color is red");
      break;
   case "green":
      Console.WriteLine("color is green");
      break;
```

```
    default:
        break;
}
```

5.

Let's turn this into something a user can make use of. Swap out `"Horror"` for `Console.ReadLine()` to get the user's response and save it to `genre`. Before that, add a `Console.WriteLine()` that prompts the user to pick a genre.

Type `dotnet run` into the terminal to see the program in action.

Hint

To get user input, use the command `Console.ReadLine()` and save the response to a `string` variable:

```
Console.WriteLine("Enter your username.");
string userName = Console.ReadLine();
```