Practical Task 1.1

(Pass Task)

General Instructions

This practical task introduces you to ***selection*** and ***casting***.

Selection allows us to make decisions in the programmes that we write. You are likely familiar with the standard ***if statement*** that is commonly used in programming languages. Though it is still appropriate to use the ***if statement*** in most of situations to solve a problem at hand, the ***switch statement*** is a powerful alternative with a *number of advantages*; it is a valuable instrument in practice, especially in developing event-driven applications. Furthermore, it is easy to get lost in the curly braces of a nested ***if statement***; so, one has to always look out for this when debugging.

When we deal with a variable saved as a particular data type, we may have to change the type to allow us to work with the variable. This operation is ***casting***. A simple example considers reading in text from the console using the `Console.ReadLine()` command, which in C# returns data as a string. How can we proceed with a given data when we asked for a user's age? In this case, we need to cast (or convert) the original input to an *integer* or *double*.

1. Create a new Console Application project and write a program to print a number to the console as a word. This program is to replace for you the traditional 'Hello World' example. It is required to accept an integer value as the input (i.e., numbers 1 to 9), and then print the corresponding word on the screen (i.e., "ONE", "TWO", ... , "NINE"). If anything other than 1 to 9 is entered, then an error message must be displayed. You may wish to implement the ***try-catch statement*** (if you know how to do this), but it is not mandatory in this task and the use of the ***if statement*** is enough. This simple program is used to make you thinking about validation; that is, has the user input only integer values between 1 and 9? If not, then we need to output the error message.

   Give a new name to the main class by renaming the default `Program` class name (and the associated file name) to `IfStatement` in the Solution Explorer of your IDE (Microsoft Visual Studio or Studio Code). Type the following code into the main `IfStatement` class and complete the other ***else if*** statements.

```
static void Main(string[] args)
{
    Console.WriteLine("Enter the number (as an integer): ");
    int number=Convert.ToInt32(Console.ReadLine());
    if (number == 1)
    {
        Console.WriteLine("ONE");
    }
    else if (number == 2)
    {

    }
```

   Complete the missing parts of the program for the remaining numbers 2 to 9. Compile and run the program, then examine the produced output. You should aim to test your program with the following:

   - All nine integers between 1 and 9.
   - A few other integers outside this range, i.e. 10 and 100.
   - What happens if you enter -5?
   - What happens if you enter a letter, e.g. 'a'? How can you add an error message to catch this? (HINT: We normally use a ***try-catch block*** to catch the error. Do not worry if you are unable to complete this now, we will look at it in more detail in the unit later.)

2. Now, create a new project and name its main class (and file) as `SwitchStatement`. Implement another version of the above program by writing the following code.

```
Console.WriteLine("Enter a number (as an integer): ");
int number = Convert.ToInt32(Console.ReadLine());

switch (number)
{
    case 1: Console.WriteLine("One"); break;
    case 2: Console.WriteLine("Two"); break;

    default: Console.WriteLine("Error: you must enter an integer between 1 and 9"); break;
}

Console.ReadLine();
```

Complete the rest of the case statements up to 9 (between case 2 and default). Run the program and check whether its output is as expected. Test it with the following:

- All nine integers between 1 and 9.
- A few other integers outside this range, i.e. 10 and 100.
- What happens if you enter -5?
- What happens if you enter a letter, e.g. 'a'? How can you add an error message to catch this?

3. Find all existing issues with the code snippets presented below. It is highly important to try working this out on paper by tracing through the statements and then input the code into your IDE. Run it to see if you are correct, and then see if you can fix the code.

```
int number = 50;
if (number == 50) ; {
   Console.WriteLine("Number is 50");
}
```

```
int number = 60;
if (number >== 50 and number <== 100) {
   Console.WriteLine("Number is more than or equal to 50 and less than or equal to 100");
}
```

```
public class Score {
   public static void main(String[] args) {
        double score = 40;
        if score > 40{
             Console.WriteLine("You passed the exam!");
        } else score < 40{
             Console.WriteLine("You failed the exam!");
        }
   }
}
```

```
Switch (n) {
   case 1: Console.WriteLine("The number is 1");
   case 2: Console.WriteLine ("The number is 2"); break;
   default: Console.WriteLine ("The number is not 1 or 2");
   break;
}
```

```
switch (n) {
   Case 1: Console.WriteLine ("A"); break;
   case2: Console.WriteLine ("B"); break;
   Default: Console.WriteLine ("C"); break;
}
```

4. What is the output of the following code fragments? Remember to try working this out on paper by tracing through the statements and then input the code into your IDE. Run it to see if you are correct, and then see if you can fix the code.

```
int height = 13;
if ( height <= 12 )
```

```
    Console.WriteLine("Low bridge: ");
    Console.WriteLine ("proceed with caution.");
```

```
int sum = 21;
if ( sum != 20 )
    Console.WriteLine ("You win ");
else
    Console.WriteLine ("You lose ");
    Console.WriteLine ("the prize.");
```

```
int sum = 7;
if ( sum > 20 ) {
    Console.WriteLine ("You win ");
} else {
    Console.WriteLine ("You lose ");
}
    Console.WriteLine ("the prize.");
```

5. A microwave oven manufacturer recommends that when heating two items you should add 50% to the heating time, and when heating three items you should double the heating time. Heating more than three items at once is not recommended.

   Create a new project and name its main class (and file) as `Microwave`. Write a program that asks the user for the number of items and the single-item heating time. Perform the calculation, and print out the recommended heating time.

6. This exercise requires you to cast the result of an operation performed on two ***integers*** to a ***double***. You should have already used casting to convert a ***string*** to an ***int***. via

   ```
   Convert.ToInt32(Console.ReadLine());
   ```

   as a combination of the two commands. The compiler will not let you assign a value to a variable if it is a wrong type – even if that variable can hold the value. When you use casting, you are essentially making a promise to the compiler that you know the types are different and that in this particular instance it is alright for C# to push data into a new variable.

   Create a new project and name its main class (and file) as `DoCasting`. Write a program by following the steps below:

   - Declare and initialise variables for two integer values **`sum`** and **`count`**.
   - Initialise these variables to 17 and 5 respectively.
   - Declare an **`integer`** variable called **`intAverage`**.
   - Calculate the integer average, **`intAverage`**, using **`sum/count`**.
   - Print out the integer average, **`intAverage`** – is this correct?
   - Declare a **`double`** variable called **`doubleAverage`**.
   - Now try calculating the double average, **`doubleAverage`**, using **`sum/count`** – what is the result? Is this correct?
   - Cast the **`sum`** variable to a double using **`(double)sum`** and divide this by **`count`**, and print the result – is this correct?

Remember that the process for casting is to place the data type in parentheses before the operation that is being performed, which in this case is **`sum/count`**.

Further Notes

- You can familiarize yourself with the principles and implementation aspects of ***if*** and ***switch statements*** by reading Sections 2.2.1 and 2.2.2 of SIT232 Workbook available in CloudDeakin Learning Resources. You can get more details about casting in C# by exploring Sections 1.6 and 3.6 of the workbook.
- Access more code examples by exploring the following links:
  - https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/if-else

  - https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch
  - https://www.geeksforgeeks.org/switch-vs-else/
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/casting-and-type-conversions

- The outline of how to debug your C# program code is given in Section 2.10 of SIT232 Workbook. More details are available online. For example, explore
  - https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-debugger?view=vs-2019
  - https://docs.microsoft.com/en-us/dotnet/core/tutorials/with-visual-studio-code

  for Microsoft Visual Studio and Visual Studio Code, respectively.
- If you seriously struggle with the remaining concepts used in this practical task, we recommend you to start reading the entire Sections 1 and 2 of SIT232 Workbook.
- In this unit, we will use Microsoft Visual Studio 2019 to develop C# programs. Find the instructions to install the community version of Microsoft Visual Studio 2019 available on the SIT232 unit web-page in CloudDeakin in Learning Resources Software Visual Studio Community 2019. You however are free to use another IDE, e.g. Visual Studio Code, if you prefer that.

Submission Instructions and Marking Process

To get your task completed, you must finish the following steps strictly on time.

- Make sure that your programs implement the required functionality. They must compile and have no runtime errors. Programs causing compilation or runtime errors will not be accepted as a solution. You need to test your programs thoroughly before submission. Think about potential errors where your programs might fail.
- **Submit** the expected code files as a solution to the task via OnTrack submission system. You must record a short video explaining your work and solution to the task. Upload the video to one of accessible resources and refer to it for the purpose of marking. You must provide a working link to the video to your marking tutor in OnTrack.

  For this task, in your video recording, you must **demonstrate how you run your program in the debug mode** and how you can use it to detect and fix potential errors. Note that debugging is an essential skill necessary to implement errorless code: The sooner you start debugging your programs the sooner you will become proficient with programming and understanding the theory part. We do not expect you to demonstrate this skill for every program you wrote in this task; you only need to select the most difficult program as an example. In the video, complement the debugging process with your verbal explanation.
- Once your solution is accepted by the tutor, you will be invited to **continue its discussion and answer relevant theoretical questions** through the Intelligent Discussion Service in OnTrack. Your tutor will record several audio questions. When you click on these, OnTrack will record your response live. You must answer straight away in your own words. As this is a live response, you should ensure you understand the solution to the task you submitted.

  Answer all additional questions that your tutor may ask you. Questions will cover the lecture notes; so attending (or watching) the lectures should help you with this **compulsory** discussion part. You should start the discussion as soon as possible as if your answers are wrong, you may have to pass another round, still before the deadline. Use available attempts properly.

Note that we will not accept your solution after **the submission deadline** and will not discuss it after **the discussion deadline**. If you fail one of the deadlines, you fail the task and this reduces the chance to pass the unit. Unless extended for all students, the deadlines are strict to guarantee smooth and on-time work throughout the unit.

Remember that this is your responsibility to keep track of your progress in the unit that includes checking which tasks have been marked as completed in the OnTrack system by your marking tutor, and which are still to be finalised. When marking you at the end of the unit, we will solely rely on the records of the OnTrack system and feedback provided by your tutor about your overall progress and the quality of your solutions.