Lab 2: Pokemon class

You are probably familiar with Pokémon characters, at least with some of Generation 1 Pokémon. If not, you can find a list of Pokémon at http://pokemondb.net/pokedex/national. You will see that each Pokémon has a **number** and one or two **elemental types** (grass, bug, flying, etc.).

Using **Visual Studio 2017**, create an **empty project** named **Project**. Write the **definition** and **implementation** of the class **Pokemon** in **two different files** (**separate compilation**), **Pokemon.h** and **Pokemon.cpp**. Your class must have the following attributes:

- **Member variables**
  - A **string** that stores the name of the Pokémon
  - An **int** to store the Pokémon's number
  - A **string** to store the Pokémon's first type
  - A **string** to store the Pokémon's second type—if the Pokémon has no second type, the string will remain empty.
- **Default constructor**
- **Two overloaded constructors**
  - **Parameters** (in this order): the name, the number, the first type
  - **Parameters** (in this order): the name, the number, the first type, the second type
- Functions **getType1**, **and getType2**
  - Each function returns the value of the appropriate type.
- Function **commonType**
  - **Parameter:** An object of the class **Pokemon**
  - The function compares the types of the **calling object** and the **parameter object**. If at least one type is the same, the function returns true; if the objects do not have any type in common, the function returns false.
  - Note that the type order does **not** matter:

Bulbasaur: Grass **Poison** Nidoking: **Poison** Ground	They have a common type: **Poison**

- Function **print**
  - Prints information in this format ➔ Bulbasaur: Grass Poison
  - If the Pokémon has no second type, then the function prints ➔ Charmander: Fire
- **Destructor**

Use the **Main.cpp** file given to test your functions.

Do **NOT** write any implementation in the class **interface.** All **implementation** should be written in the **.cpp** file (**separate compilation**).

Make sure to:

- Add a **name header** with your name, date, etc.
- Pass by **reference** when needed and add the **const** modifier to the parameters **ONLY when necessary**.
- Do **NOT** use a **return** statement without returning anything! ➔ `return;`
- Do **NOT** use the **break** and **continue** statements (there are no switch statements to use **break**).
- Do **NOT** use global variables **ever**.
- Do **NOT** modify any of the given code.
- Use the **const** modifier when necessary for **member functions**.

Keep in mind the following:

- Divide your code in meaningful blocks for readability
- Name your variables using descriptive names
- Use all appropriate conventions for naming
- Do not leave unnecessary spaces or lines in your code

Below you can find a segment of the expected output: