# Worksheet 1.8

This worksheet focuses on developing your understanding of class inheritance in C++. A key aspect of this is being able to recognise when inheritance is a useful tool for a given problem. This worksheet also provides you with some experience of using virtual functions to override behaviour in subclasses. You will be provided with all the resources you need to complete this worksheet.

**Please ensure that you create a header file (.h) with header guard (or #pragma once directive) and code file (.cpp) for each class**

## Task 1: carbon footprints

Governments and companies worldwide are becoming increasingly concerned with carbon footprint (annual release of carbon dioxide into the atmosphere) from buildings burning various types of fuels for heat, vehicles burning fuels for power, and the like. Many scientists blame these greenhouse gases for the phenomenon called global warming. This task involves creating several classes to model this phenomenon using inheritance and polymorphism.

Create three classes (Building, Car and Bicycle) which are initially unrelated by inheritance. Give each class some appropriate attributes and behaviour that it does not have in common with the other classes.

Create an abstract class called CarbonFootprint with only one pure virtual function called getCarbonFootprint().

Make each of your classes inherit from the CarbonFootprint class and implement a getCarbonFootprint() function for each class. Use the internet to find a realistic carbon footprint value for each of them.

Write a small application that creates an instance of each class. Create a vector of pointers to CarbonFootprint objects and add the address of each instance to the vector. Next, iterate over the vector (using the at() function rather than the [] operators) and polymorphically invoke the getCarbonFootprint() function and output the objects carbon footprint.

## **Task 2: logically that might be true**

The following UML Class Diagram provides an Object-orientated design for a simple logic-based system capable of performing AND, OR and NOT operations:

Diagram

Description automatically generated

This design makes use of inheritance and abstract classes to provide a generic approach to this problem. It would be easy to add extra logic gates to this system such as XOR, NAND or NOR. This task involves creating a set of classes based on this class diagram using single inheritance and pure virtual functions.

Write a small application that makes use of the new classes to print out the truth table for the following logic circuit:

Diagram, schematic

Description automatically generated

Now, add a new XORGate class to your system which inherits from BinaryGate and implement an XOR operation. Use your new XORGate class to validate the truth table you printed out for the logic circuit above.

## **JOURnal Task 7**

**Your task is to complete Task 2 (as detailed above). Your Journal entry should reflect on your experiences in completing this task. This will include a short explanation of what you did, any problems encountered and important discoveries, and a compressed copy of your project.**

**NOTE: Please name your journal entry for this task as TASK 7.**