Work Schedule

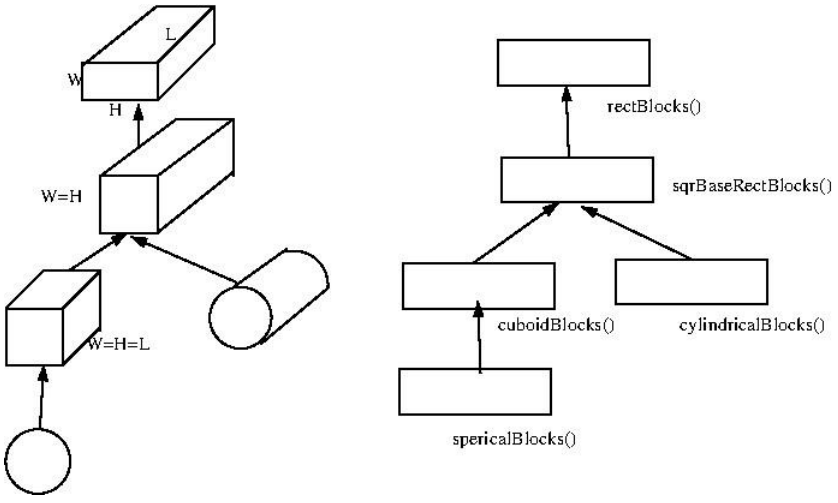
Problem Description

The dimensions of wooden blocks are given in a data file, *dataBlocks.dat* and gives for each block the width (W) and height (H) of its base and the length (L) of the block. The problem is simply to read these blocks and store them first as an array of blocks of rectangular objects of a class rectBlocks().

From the class of rectangular blocks, there are some whose base width equals the base height (i.e., W = H). Call these the square base rectangles, sqrBaseRectBlocks(). Generate an array of only the blocks with square bases as a derived class of rectBlocks().

From the class of sqrBaseRectBlock(), two other classes may be derived; one with W = H = L, call this the cuboidBlocks() and another, formed as cylindrical blocks, where the square base is formed into a circle with diameter the same as the width or height of the square base. Call these objects of the cylindricalBlocks() class.

From the objects of the cuboidBlocks() class, we now derive spherical objects, sphericalBlocks(), whose diameters equal the width, height or length of the respective cuboid objects they are derived from. The hierarchy of class dependencies may be depicted as illustrated in the Figure **??**.



Tasks to be done

A) Write the class rectBlocks(), as a base class and create the array (vector) of objects of this class using the data from the input file *dataBlocks.dat*

B) Construct the other appropriate classes following the illustrated class hierarchy diagram.

C) Create the array of objects of the sphericalBlocks(). Sort the array in ascending order of their diameters and print in ascending order of their diameters for the diameter, spherical surface and volume of the objects.

D) Create a similar array of objects of the cylindricalBlocks() class. Sort the array in ascending order of their cylindrical surface area and print in ascending order of cylindrical surface area, the diameter of the circular base, the length of the object and the area.

E) Organise your application into appropriate functions and compilation units and integrate these with a makefile. Decomposing the work into functions and dependent compilable units is part of the exercise.

F) Please include the make file for running your program