CS1020E | PE 1 | Problem 2 (50 marks)

Shipping Books

Problem Description

You are running an online book store. Each day at the warehouse, you will pack books ordered by each customer into one or more cuboid-shaped packages and send them to your customers via a shipping company.

Each book is also cuboid-shaped, with length, width, and height, all measured in meters (m). The length runs along the book spine, and the height is the thickness of the book. The width may be longer than the length for some books. The mass of the book is measured in kilograms (kg).

You have a ***standard packing method*** to pack a set of books into a package. You always stack the books cover-to-cover, and with all the spines parallel to each other. Then you will make a packaging box with height exactly the same as the total height (thickness) of the books. The length of the box will be equal to the largest length of the books, and the width will be equal to the largest width of the books. The package's actual mass is the total mass of the books in it.

Your shipping company is using the concept of "*volumetric mass*" to charge you. For each package, the ***billable mass*** is the higher of the following two:

- the volume of the package in m^3 times 500 kg/m^3, or
- the actual mass of the package.

The ***shipping charge*** for each package is calculated as follows:

1) The billable mass is rounded up to the nearest whole kg.
2) For the first kg — \$1.00.
3) For the second kg — \$0.80.
4) For the third kg — \$0.60.
5) Subsequent kg — \$0.40 per kg.

For example, for a package with billable mass of 4.5 kg, the shipping charge will be \$1.00 + \$0.80 + \$0.60 + 2 × \$0.40 = \$3.20.

Given a set of books to be put in a package, the standard packing method described above often does not produce the minimum package volume. We want to minimize the package volume by allowing some books, if necessary, to be rotated 90 degrees such that their spines run along the width of the packaging box. This is the ***improved packing method***. The longer side of the resulting package box is considered the length and the shorter side the width.

You are required to complete a C++ program that takes in a set of books as input and produces the required output as described below.

Add your code only to the parts of the files indicated. Do not modify any other part of the given code, and do not add new file, remove file or rename any file.

Fill in your particulars (name, NUSNET ID, plab userID) at the beginning of every file.

Input

- The first line of the input contains one integer *N*, where $N > 0$.
- It is then followed by *N* lines, where each line contains the *title*, *length*, *width*, *height*, and *mass* of a book. The *title* is a word with no whitespace, and the *length*, *width*, *height*, and *mass* are non-integer numbers.

Output

The output has the following 8 lines:

```
Package standard length (m): x1
Package standard width (m): x2
Package height (m): x3
Package mass (kg): x4
Billable mass (kg): x5
Shipping charge ($): x6
Package improved length (m): x7
Package improved width (m): x8
```

- ***x1*** is the package length produced using the standard packing method.
- ***x2*** is the package width produced using the standard packing method.
- ***x3*** is the package height.
- ***x4*** is the package mass.
- ***x5*** is the billable mass of the package produced using the standard packing method.
- ***x6*** is the shipping charge for the package produced using the standard packing method.
- ***x7*** is the package length produced using the improved packing method.
- ***x8*** is the package width produced using the improved packing method.

All numbers must be output with two decimal places after the decimal point.

Sample Input 1

```
2
BookX 0.30 0.20 0.06 1.5
BookY 0.25 0.40 0.12 3.4
```

Sample Output 1

```
Package standard length (m): 0.30
Package standard width (m): 0.40
Package height (m): 0.18
Package mass (kg): 4.90
Billable mass (kg): 10.80
Shipping charge ($): 5.60
Package improved length (m): 0.40
Package improved width (m): 0.25
```

Sample Input 2

```
3
Book1 0.30 0.20 0.04 1.5
Book2 0.25 0.30 0.05 2.0
Book3 0.35 0.20 0.03 1.2
```

Sample Output 2

```
Package standard length (m): 0.35
Package standard width (m): 0.30
Package height (m): 0.12
Package mass (kg): 4.70
Billable mass (kg): 6.30
Shipping charge ($): 4.00
Package improved length (m): 0.35
Package improved width (m): 0.25
```

Skeleton Code

```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include <cmath>    // for the ceil() function.
using namespace std;


class CuboidObject
{
protected:
    double _length, _width, _height, _mass;
public:
    CuboidObject() {}
    CuboidObject( double length, double width,
                  double height, double mass )
      : _length(length), _width(width), _height(height), _mass(mass) {}

    double getLength() { return _length; }
    double getWidth()  { return _width; }
    double getHeight() { return _height; }
```

```
    double getVolume() { return _length * _width * _height; }
    double getMass()   { return _mass; }
};


class Book : public CuboidObject
{
private:
    string _title;
public:
    Book( string title, double length, double width,
          double height, double mass )
      : CuboidObject( length, width, height, mass ), _title(title) {}

    string getTitle() { return _title; }
};


class Package : public CuboidObject
{
private:
    vector<Book*> _books;

    // _improvedLength and _improvedWidth contain the length and
    // width of the package when the improved packing method is used.
    double _improvedLength, _improvedWidth;

    // _length and _width from CuboidObject contain the length and
    // width of the package when the standard packing method is used.

    void _computeStandardLengthWidth();
    void _computeImprovedLengthWidth();

public:
    Package( vector<Book*> &books );
    vector<Book*>& getBooks() { return _books; }
    double getImprovedLength() { return _improvedLength; }
    double getImprovedWidth()  { return _improvedWidth; }
};


// Constructor of Package
Package::Package( vector<Book*> &books ) : _books(books)
{
    // Write your code here.
}


// Sets _length and _width to the length and width of the package
// produced using the standard packing method.
void Package::_computeStandardLengthWidth()
{
    // Write your code here.
}
```

```
// Sets _improvedLength and _improvedWidth to the length and width
// of the package produced using the improved packing method.
void Package::_computeImprovedLengthWidth()
{
     // Write your code here.
}


// Returns the billable mass of the input package.
double computeBillableMass( Package &package )
{
    const double volMassFactor = 500.0;

    // Write your code here.
}


// Returns the shipping charge given the billable mass.
double computeShippingCharge( double billableMass )
{
    // Write your code here.
}


int main()
{
    vector<Book*> books;

    // Write your code here to read the inputs.


    // Outputs
    Package package( books );
    double billableMass = computeBillableMass( package );
    double shippingCharge = computeShippingCharge( billableMass );

    cout << setprecision(2) << fixed;
    cout << "Package standard length (m): "
         << package.getLength() << endl;
    cout << "Package standard width (m): " << package.getWidth() << endl;
    cout << "Package height (m): " << package.getHeight() << endl;
    cout << "Package mass (kg): " << package.getMass() << endl;
    cout << "Billable mass (kg): " << billableMass << endl;
    cout << "Shipping charge ($): " << shippingCharge << endl;

    cout << "Package improved length (m): "
         << package.getImprovedLength() << endl;
    cout << "Package improved width (m): "
         << package.getImprovedWidth() << endl;

    // Not required to deallocate dynamically allocated memory.

    return 0;
}
```

Submission

Just leave your completed program file in the **skeleton/** directory. There is no need to submit it to CodeCrunch.