For hierarchy of shapes discussed in class, assume Rectangle class has a method draw given below. Write down the similar methods for square and triangle classes. [Note: data members of rectangle class are point2d topleft, int length, int width and that of triangle are v1, v2 and v3 of type point2d]

```
void draw()
{
        point2d p1(this->topleft);
        point2d p2(this->topleft.getx(), this->topleft.gety()+this->length);
        point2d p3(this->topleft.getx()+this->width, this->topleft.getyy());
        point2d p4(this->topleft.getx()+this->width,
                                this->topleft.gety()+this->length);

        drawline(p1, p2);
        drawline(p1, p3);
        drawline(p2, p4);
        drawline(p3, p4);
}
```

Consider following description type **matrix**, describe class **squarematrix** inheriting from class matrix. The **squarematrix** class should reuse maximum of the matrix class functionality and may override or extends it. [Hint: it may override the transpose method with an efficient one; it may have methods for determinant and symmetric tests, etc]

```
class matrix
{
  protected:
    int rows, cols;
    double **data;
  public:
    matrix(); // 0,0 for rows and colums
    matrix(int, int); // rows and columns
    matrix(const matrix &); //copy constructor
    virtual ~matrix();
    matrix operator=(const matrix &);
    bool operator==(const matrix &);
    bool operator!=(const matrix &);
    overloading of arithmetic operators
    overloading of input and output operators
    virtual matrix transpose() const;
};
```

Using polymorphism, you are required to implement the inheritance hierarchy shown in figure to compute percent marks of various types of **Student**s.

There is one base class (**Student**) to store common data, and three derived classes that divide the set of students into three categories: English students (**EngStudent**), Computer Science students (**CompStudent**), and Math students (**MathStudent**). All the data members of the **Student** class should be declared as *protected*. Note that the member variables of each of the three derived classes are not shown in the above class diagram. Each derived class will have appropriate number of *private* data members to store the marks of different assessments applicable for a particular student. Assessments and their weightages for the three categories of students are as follows:

- **English** – Attendance = 10%, Report = 30%, Midterm = 30%, Final Exam = 30%
- **Computer Science** – Project = 25%, Midterm = 35%, Final Exam = 40%
- **Math** – Quiz Average* = 25%, Homeworks = 25%, Final Exam = 50%

Note the following important points:

1. Student class must define float percentMarks() as pure virtual
2. You may have to create a class ExamSys to aggregate Students
3. You MUST have to write driver code (or main function) for sufficient hard coded student's data