design a set of classes for storing student information, along with a class that will manage a list of students. Data can be imported from files for storage in the list, and summary reports with computed final grades will be printed to output files.

Details

1. Design a set of classes that store student grade information. There should be one base class to store common data, and three derived classes that divide the set of students into three categories: English students, History students, and Math students. All **data** stored in these classes should be private or protected. Any access to class data from outside should be done through public member functions. The **base** class should allocate storage for the following data (and only this data):
    - student's first name (you may assume 20 characters or less)
    - student's last name (you may assume 20 characters or less)
    - Which course the student is in (English, History, or Math)

2. Each class should have a function that will **compute** and **return** the student's final average, based on the stored grades. All grades are based on a 100 point scale. Here are the grades that need storing for each subject, along with the breakdown for computing each final grade:

    English -- Attendance = 10%, Project = 30%, Midterm = 30%, Final Exam = 30%

    History -- Term Paper = 25%, Midterm = 35%, Final Exam = 40%

    Math -- There are 5 quizzes, to be averaged into one Quiz Average (which can be a decimal number). Final grade computed as follows:
    * Quiz Average = 15%, Test 1 = 25%, Test 2 = 25%, Final Exam = 35%

3. Write a class called `StudentList`, which will be used to store the list of various students, using a single array of flexible size. Note that this array will act as a heterogeneous list, and it will need to be dynamically managed. Each item in the array should be a Student pointer, which should point to the appropriate type of object. Your list should only be big enough to store the students currently in it. Your StudentList class needs to have these public functions available:
    - **`StudentList()`** -- default constructor. Sets up an empty list
    - **`~StudentList()`** -- destructor. Needs to clean up all dynamically managed data being managed inside a StudentList object, before it is deallocated
    - **`bool ImportFile(const char* filename)`**

        - This function should add all data from the given file (the parameter) into the internally managed list of students. The file format is specified below in this writeup. (Note that if there is already data in the list from a previous import, this call should add MORE data to the list). Records should be added to the end of the given list in the same order they appear in the input file. If the file does not exist or cannot be opened, return false for failure. Otherwise, after importing the file, return true for success.
    - **`bool CreateReportFile(const char* filename)`**
        - This function should create a grade report and write it to an output file (filename given by the parameter). Grade report file format is described below in this writeup.
          If the output file cannot be opened, return false for failure. Otherwise, after writing the report file, return true for success.
    - **`void ShowList() const`**
        - This function should print to the screen the current list of students, one student per line. The only information needed in this printout is last name, first name, and course name. Format this output so that it lines up in columns

Note that while this class will need to do dynamic memory management, you do **not** need to create a copy constructor or assignment operator overload (for deep copy) for this class. (Ideally we would, but in this case, we will not be using a StudentList object in a context where copies of such a list will be made) with the appropriate prototypes

4. Write a main program (in a separate file) that creates a single StudentList object and then implements a menu interface to allow interaction with the object. Your main program should implement the following menu loop (any single letter options should work on both lower and upper case inputs):

```
                  *** Student List menu ***

         I       Import students from a file
         S       Show student list (brief)
         E       Export a grade report (to file)
           M       Show this Menu
           Q       Quit Program
```

The import and export options should start by asking for user input of a filename (you may assume it will be 30 chars or less, no spaces), then call the appropriate class function for importing a file or printing the grade report to file, respectively. (If the inport/export fails due to bad file, print a message indicating that the job was not successfully completed).

The "Show student list" option should call the class function that prints the brief student list to screen (names and course info only).

The Show this Menu option should re-display the menu.

Quit should end the menu program. (Until this option is selected, keep looping back for menu selections).

12. **File formats**

Input File -- The first line of the input file will contain the number of students listed in the file. This will tell you how many student records are being imported from this file. After the first lines, every set of two lines constitutes a student entry. The first line of a student entry is the name, in the format *lastName, firstName*. Note that a name could include spaces -- the comma will delineate last name from first name. The second line will contain the subject ("English", "History", or "Math"), followed by a list of grades (all integers), all separated by spaces. There will be no extra spaces at the ends of lines in the file. The order of the grades for each class type is as follows:

English -- Attendance, Project, Midterm, Final Exam
History -- Term Paper, Midterm, Final Exam
Math -- Quiz 1, Quiz 2, Quiz 3, Quiz 4, Quiz 5, Test 1, Test 2, Final Exam

Output File -- The output file that you print should list each student's name (*`firstName lastName`* - no extra punctuation between), Final Exam grade, final average (printed to 2 decimal places), and letter grade (based on 10 point scale, i.e. 90-100 A, 80-89 B, etc). Output should be separated by subject, with an appropriate heading before each section, and each student's information listed on a separate line, in an organized fashion. (See example below). The order of the students within any given category should be the same as they appear in the student list. Data must line up appropriately in columns when multiple lines are printed in the file. At the bottom of the output file, print a grade distribution of ALL students -- how many As, Bs, Cs, etc.

13. **General Requirements**
    - No global variables, other than constants!
    - All member data of your classes must be private or protected
    - Use the `const` qualifier on member functions wherever it is appropriate.
    - The code for this program should be portable. Test with g++ compiler commands before submitting
    - You may use any of the standard I/O libraries that have been discussed in class (`iostream`, `iomanip`, `fstream`, as well as C libraries, like `cstring`, `cctype`, etc). You may also use the `string` class library
    - You may not use any of the other STL (Standard Template Libraries) besides `string`
    - Do not use any C++11-only libraries or features