Due date: Feb 16, 11:59 PM

Homework 1

1. Implement a Student class.
    a. Create a class Student with the following private data members:
        1. name
        2. exam_1 grade
        3. exam_2 grade
    b. Create all appropriate accessor and mutator functions.
    c. Assign appropriate access modifiers to insure encapsulation.
    d. Add a private calcGPA() function that calculates and returns the GPA based upon the two exam grades.
    e. Add a public getGrade() function that:
        1. Obtains the GPA from the private calcGPA() function.
        2. Returns a letter grade based upon the numerical GPA value.
        90 to 100 = A
        80 to 90 = B
        70 to 80 = C
        60 to 70 = D
        0 to 60 = F
    f. Test all functions use following main.

```
int main(){
  Student a;
  a.setName("David");
  a.setExam1(90);
  a.setExam2(80);
  cout<<a.getName()<<endl;
  cout<<a.getExam1()<<endl;
  cout<<a.getExam2()<<endl;
  cout<<a.getGrade()<<endl;
}
```

2. Using the class from problem 1, replace main with the following:
    a. Implement a partially filled array of type Student named students of capacity 10.
    b. Implement a non-member addStudent() function that:
        1. Creates a new student with data populated by input parameters.
        2. Adds the new student to the students array.
    c. Implement a non-member output() function that:
        1. Outputs all student data in the students array as displayed in the output example (see next page).
    d. Main should use the addStudent and output functions to create five students and display their content to the console (see next page).

    f. Test all functions use following main.

Due date: Feb 16, 11:59 PM

```
int main(){
  int capacity = 10;
  Student students[capacity];
  int num = 0;
  addStudent(students,capacity,num,"Amy",95,90);
  addStudent(students,capacity,num,"Bob",74,63);
  addStudent(students,capacity,num,"Charlie",86,80);
  addStudent(students,capacity,num,"Daisy",75,90);
  addStudent(students,capacity,num,"Edward",24,66);
  output(students,num);
}
```

Output Example

```
Name: Amy
Exam 1: 95
Exam 2: 90
GPA: A

Name: Bob
Exam 1: 74
Exam 2: 63
GPA: D


Name: Charlie
Exam 1: 86
Exam 2: 80
GPA: B

Name: Daisy
Exam 1: 75
Exam 2: 99
GPA: B

Name: David
Exam 1: 24
Exam 2: 66
GPA: F
```

Due date: Feb 16, 11:59 PM

3. N-Queens.

A Queen on a chessboard can attack any piece in the same column, row or diagonal. The N-Queens problem is to place n queens on a n x n chessboard such that no two queens threaten each other.

a) Implement a one-dimensional integer array of Queen positions for an 8x8 board where indices represent rows and the values represent columns.

For example, this “safe” solution would be {3,6,2,7,1,4,0,5}

```
. . . Q . . . .
. . . . . . Q .
. . Q . . . . .
. . . . . . . Q
. Q . . . . . .
. . . . Q . . .
Q . . . . . . .
. . . . . Q . .
```

b) Request values for the array from the console.
c) Implement an output to display the board (see output example).
d) Implement a queensAreSafe function that:
    1) Returns false if multiple queens share a column. Note that by design they are in separate rows (make sure you understand why).
    2) Returns false if multiple queens share a diagonal.
    3) Returns true if all queens are safe.
e) Program should display if the Queens are safe or not safe.

Example output (input is bold and italicized):
Enter 8 column values: **1 4 2 3 5 7 6 0**

```
. Q . . . . . .
. . . . Q . . .
. . Q . . . . .
. . . Q . . . .
. . . . . Q . .
. . . . . . . Q
. . . . . . Q .
Q . . . . . . .
```

Queens are not safe!

(**3,6,2,7,1,4,0,5, safe figure**)

Due date: Feb 16, 11:59 PM

4. Implement a class name Vehicle:
    a. A Vehicle class with two private data members: brand, number_of_doors
    b. Implement two arguments constructor
    c. Implement default constructor and with constructor delegation, initialize with "TBD", and 0.
    e. Vehicle class:
        1) All appropriate accessor and mutator functions.
        2) Implement a **display** member function that prints object data as demonstrated in the output example.

Output Example:

```
Brand: TBD
Number of Doors: 0
Brand: BMW
Number of Doors: 4
```

Use following main to test your class.

```
int main() {
   Vehicle a,b("BMW", 4);
   a.display();
   b.display();

   cout << endl;
   return 0;
}
```

5. Implement a class:
    a. A Student class with a private nested GPA class and three data members: name, major, grade (grade is of type GPA)
    b. Use c++11 member initialization to set the default name and major to "blank" and GPA to 0.0 for all objects.
    c. Implement multiple constructors with constructor delegation.
    d. GPA class:
        1) All appropriate accessor and mutator functions.
        2) Implement a getLetterGrade function which returns a letter based upon the GPA as such: >=3.5 A, >=2.5 B, >=1.5 C, >=1 D, all else F
    e. Student class:
        1) All appropriate accessor and mutator functions.
        2) Apply the const member function modifier as needed.
        3) Implement a **display** member function that prints object data as demonstrated in the output example.

    Instantiate two objects and display their data as such:

Due date: Feb 16, 11:59 PM

Output Example

```
Name: John Williams
Major: Music
GPA: 4.00
Grade: A
Name: Isaac Asimov
Major: English
GPA: 2.53
Grade: B
```

Use following main to test your class.

```
int main() {
    cout.setf(ios::fixed); // set to print two digit after decimal
    cout.precision(2);

    Student s1("John Williams", "Music", 4.0);
    Student s2("Isaac Asimov", "English", 2.53);
    s1.display();
    s2.display();

    cout << endl;
    return 0;
}
```

Grading policy:

Should submit .cpp file format, other format will be not accepted and assigned 0 point directly.

50% points loss if the program doesn't compile. 50% points for the rest.
If the code compiles and runs, full points if it succeeds for all requirements.