course.h

Key

```
1   #include <string>
2
3   using namespace std;
4
5   struct course {
6       string course_name;
7       int num_credits;
8       string term;
9       string grade;
10  };
11
```

Student.h

```
1   #include "course.h"
2
3   using namespace std;
4
5   class Student {
6       private:
7           int id_num;
8           string first_name;
9           string last_name;
10          string major;
11          float gpa;
12          int num_credits;
13          int num_courses;
14          course* courses;
15          static int count;
16      public:
17          //constructors
18          Student();
19          Student(int, string, string, string, float, int, int);
20          //accesors
21          int get_id_num() const;
22          string get_first_name() const;
23          string get_last_name() const;
24          string get_major() const;
25          float get_gpa() const;
26          int get_num_credits() const;
27          int get_num_courses() const;
28          string get_course_name(int) const;
29          int get_course_credit(int) const;
30          string get_course_term(int) const;
31          string get_course_grade(int) const;
32          //mutators
33          void assign_id_num(int);
34          void set_first_name(string);
35          void set_last_name(string);
36          void set_major(string);
37          void calculate_gpa();
38          void calc_num_credits();
39          void set_num_courses(int);
40          void set_course(string, int, string, string, int);
41          static int get_count();
42          //copy constructor
43          //assignment operator overload
44          //destructor
45
46  };
47
```

This includes from our directory

This is a dynamic array of courses

This is a variable shared among all objects of the class

These are all const because they are not changing the content, just returning value. This means they can be used with const objects.

Notice not all mutators need the word "set." Remember, function names are largely arbitrary. A mutator is any function which alters a private member variable.

This is a static function, to alter or work with variables shared by all objects of the class type.

→ Cover on Friday

Questions

What is line 14?

What is line 15?

Label the function categories on lines 17, 20, 32.

What is a constructor? Does this class have a default constructor?

What would happen if we declared Student a; in main() but ommitted line 18 from the class?

Why are the function on lines 21 - 31 const functions?

What is an accesor function? What is a mutator function?

Are lines 33, 37, and 38 mutator functions? What is the function on line 41?

Student.cpp

```cpp
#include <iostream>
#include "student.h"

using namespace std;

// constructors
/*****************************************************************************
** Function: Student
** Description: Default constructor
** Parameters: None
** Pre-Conditions:
** Post-Conditions: initializes the pmvs of the class when the object is created
*****************************************************************************/
Student::Student() {
        id_num = 0;                    num_courses = 0;
        first_name = "Test";
        last_name = "Student";
        major = "n/a";
        gpa = 0.0;
}       num_credits = 0;
/*****************************************************************************
** Function: Student
** Description: Non default constructor
** Parameters: int id value, string first name val, string lastname val, string major, float gpa, int num_credits, int num_courses
** Pre-Conditions: all parameters must be correctly initialized
** Post-Conditions: initializes pmvs w/ appropriate values and assigns memory to courses*
*****************************************************************************/
Student::Student(int idn, string fn, string ln, string m, float gpa, int ncr, int nc) {
      id_num = id;
      first_name = fn;
      last_name = ln;
      major = m;
      this->gpa = gpa;
      num_credits = ncr;
      num_courses = nc;
      courses = new course[num_courses];
      for(int i=0; i< num_courses; i++){
          courses[i].course_name = "Course";
          courses[i].num_credits = 0;
      }   courses[i].term = "n/a";
}         courses[i].grade = "n/a";
// accessors
int Student::get_id_num() const { return id_num; }
string Student::get_first_name() const { return first_name; }
string Student::get_last_name() const { return last_name; }
string Student::get_major() const { return major; }
float Student::get_gpa() const { return gpa; }
int Student::get_num_credits() const { return num_credits; }
int Student::get_num_courses() const { return num_courses; }
string Student::get_course_name(int index) const { return courses[index].course_name; }
int Student::get_course_credit(int index) const { return courses[index].num_credits; }
string Student::get_course_term(int index) const { return courses[index].term; }
string Student::get_course_grade(int index) const { return courses[index].grade; }

// mutators
void Student::assign_id_num(int id) { id_num = id; }
void Student::set_first_name(string name) { first_name = name; }
void Student::set_last_name(string name) { last_name = name; }
void Student::set_major(string m) { major = m; }
/*****************************************************************************
** Function: calculate gpa
** Description: derives the current gpa from the courses array and sets the gpa for the obj
** Parameters: none
** Pre-Conditions: courses needs to be allocated and initialized
** Post-Conditions: sets the gpa for the object
```

this->gpa refers to the one belonging to the obj

Inline functions

```
70  ********************************************************************/
71  void Student::calculate_gpa() {
72      float total = 0;
73      for(int i=0; i<num_courses; i++) {
74          if(courses[i].grade == "A")
75              total += (4.0 * courses[i].num_credits);
76          else if(courses[i].grade == "A-")
77              total += (3.7 * courses[i].num_credits);
78          else if(courses[i].grade == "B+")
79              total += (3.3 * courses[i].num_credits);
80          else if(courses[i].grade == "B")
81              total += (3.0 * courses[i].num_credits);
82          else if(courses[i].grade == "B-")
83              total += (2.7 * courses[i].num_credits);
84          else if(courses[i].grade == "C+")
85              total += (2.3 * courses[i].num_credits);
86          else if(courses[i].grade == "C")
87              total += (2.0 * courses[i].num_credits);
88          else if(courses[i].grade == "C-")
89              total += (1.7 * courses[i].num_credits);
90          else if(courses[i].grade == "D+")
91              total += (1.3 * courses[i].num_credits);
92          else if(courses[i].grade == "D")
93              total += (1.0 * courses[i].num_credits);
94          else
95              total += 0.0;
96      }
97      gpa = total/float(num_credits);
98  }
99  /*************************************************************************
100 ** Function: calc_num_credits
101 ** Description: updates the current number of credits based on courses
102 ** Parameters: none
103 ** Pre-Conditions: courses needs to be allocated and initalized
104 ** Post-Conditions: num_credits is updated for the ~~stu~~ obj;
105 *************************************************************************/
106 void Student::calc_num_credits() {
107     int total = 0;
108     for(int i=0; i<num_courses; i++) {
109         total += courses[i].num_credits;
110     }
111     num_credits = total;
112 }
113
114 void Student::set_num_courses(int nc) { num_courses = nc; }
115 /*************************************************************************
116 ** Function: set_course
117 ** Description: populates a course wl values
118 ** Parameters: string name of course, int num credits, string term, string grade, int index in
119 ** Pre-Conditions: courses is allocated & ~~initialized~~ array
120 ** Post-Conditions: one course in array is set to input vals
121 *************************************************************************/
122 void Student::set_course(string name, int credits, string term, string grade, int index)
    {
123     if(index < 0 || index > num_courses) {
124         cout << "Index error" << endl;
125     }
126     else {
127         courses[index].course_name = name;
128         courses[index].num_credits = credits;
129         courses[index].term = term;
130         courses[index].grade = grade;
131     }
132 }
133 int Student::count = 0;   static variable initialization, can only be done once
134 /*************************************************************************
135 ** Function: get count
136 ** Description: static function to increment and return the val of count
137 ** Parameters: none
```

```
138     ** Pre-Conditions: count intialized, objects of class exist
139     ** Post-Conditions: count incremented and value returned
140     ****************************************************************************/
141     int Student::get_count() {
142         count++;
143         return count;
144     }
145     //copy constructor
146     //assignment operator overload
147     //destructor
148
```

input.txt

```
C.J. Cregg Political_Science 4
PS201 4 F16 A
PS202 4 W17 A
PS300 4 SP17 B
COMM400 3 F16 A
Olivia Dunham Criminal_Justice 2
COMM100 3 F16 A-
CH200 4 W17 B+
Josh Lyman Law 4
PS401 4 F16 A
PS402 4 W17 A
PS403 4 SP17 A
HST501 3 F16 B+
Toby Ziegler Communications 3
COMM315 4 F16 A
ECON200 4 W17 B-
COMM400 3 SP17 A
Leslie Knope Public_and_Environmental_Affairs 3
HST350 3 W17 A
BIO111 4 W17 C+
PS201 4 W17 B+
Sam Seaborn Law 4
PS401 4 F16 A
PS402 4 W17 A
PS403 4 SP17 A
HST501 3 F16 A
Walter Bishop General_Sciences 3
PH500 4 W16 A
BIO500 4 W16 A
CH500 4 W16 A


```

First_Name Last_Name Major Num_Courses

Course_Name Num_credits Term Grade

<Repeat Num_Course times>

driver.cpp

```
1  #include <iostream>
2  #include <fstream>
3  #include "student.h"
4
5  /*************************************************************************
6  ** Function: pop from file
7  ** Description: populates an array of student objects wi information from a file
8  ** Parameters: static array of student objs, num students
9  ** Pre-Conditions: array is allocated, file is correctly formatted and exists, num students is correct
10 ** Post-Conditions: array of student obj is initialized
11 *************************************************************************/
12 void pop_from_file(Student s[], int num_students) {
13     ifstream rf; create file obj read only
14     string f_name, l_name, m, cn, term, grade;
15     int nc = 0, id, ncr =0;
16     rf.open("input.txt"); open file
17     for(int i=0; i<num_students; i++) {
18         rf >> f_name;
19         rf >> l_name;
20         rf >> m;
21         rf >> nc;
22         id =  100 + Student::get_count(); Base the id on how many student object exist
23         s[i] = Student(id, f_name, l_name, m, 0.0, 0, nc); Call non default constructor
24         for(int j=0; j<s[i].get_num_courses(); j++) {
25             rf >> cn;
26             rf >> ncr;
27             rf >> term;
28             rf >> grade;
29             s[i].set_course(cn, ncr, term, grade, j);
30         }
31         s[i].calc_num_credits();
32         s[i].calculate_gpa();
33     }
34     rf.close(); close file
35 }
36 /*************************************************************************
37 ** Function: print students
38 ** Description: prints all student information
39 ** Parameters: a student obj
40 ** Pre-Conditions: the student obj is initialized
41 ** Post-Conditions: output to the screen
42 *************************************************************************/
43 void print_students(const Student& a) {
44     cout << "Name: " << a.get_first_name() << " " << a.get_last_name() << endl;
45     cout << "ID: " << a.get_id_num() << endl;
46     cout << "Major: " << a.get_major() << endl;
47     cout << "GPA: " << a.get_gpa() << " Credits: " << a.get_num_credits() << endl;
48     cout << "Courses: " << endl;
49     for(int i=0; i<a.get_num_courses(); i++) {
50         cout << a.get_course_name(i) << " Credits: " << a.get_course_credits(i);
51         cout << " Term: " << a.get_course_term(i) << " Grade: " << a.get_course_grade(i)
           << endl;
52     }
53     cout << endl;
54
55 }
56 /*************************************************************************
57 ** Function: print
58 ** Description: prints an array of students
59 ** Parameters: a static array of student obj, num students
60 ** Pre-Conditions: student array is allocated and initialized, num students is correct
61 ** Post-Conditions: output student info
62 *************************************************************************/
63 void print(Student s[], int num_students) {
64     for(int i=0; i<num_students; i++) {
65         print_students(s[i]);
66     }
67 }
68
```

read from file

```
69  int main() {
70
71      Student students[7];
72      print(students, 7); What prints here?
73      pop_from_file(students, 7);
74      print(students, 7); What prints here?
75
76      return 0;
77  }
78
```