1. Assume that student records are implemented using the following declaration.

```
struct StudentInfo
{
   apstring name;
   int creditHours;
   double gradePoints;
   double GPA;
};
```

(a) Write function `ComputeGPA`, as started below. `ComputeGPA` should fill in the `GPA` data member for the first `numStudents` records in its `apvector` parameter `roster`. A student's GPA (grade point average) is computed by dividing `gradePoints` by `creditHours`. The GPA for a student with 0 credit hours should be set to 0.

Complete function `ComputeGPA` below. Assume that `ComputeGPA` is called only with parameters that satisfy its precondition.

```
void ComputeGPA(apvector<StudentInfo> & roster, int numStudents)
// precondition:  roster contains numStudents records,
//                0 < numStudents ≤ roster.length(), in which the
//                name, creditHours and gradePoints data members
//                have been initialized.
// postcondition: The GPA data member for the first numStudents records
//                in roster has been calculated.
```

GO ON TO THE NEXT PAGE

(b) Write function `IsSenior`, as started below. `IsSenior` should return `true` if the given student has at least 125 credit hours and has a GPA of at least 2.0; otherwise, `IsSenior` should return `false`.

For example:

student				Result of the call `IsSenior(student)`
name	creditHours	gradePoints	GPA	
King	45	171	3.8	`false` (not enough credit hours)
Norton	128	448	3.5	`true`
Solo	125	350	2.8	`true`
Kramden	150	150	1.0	`false` (GPA too low)

Complete function `IsSenior` below.

```
bool IsSenior(const StudentInfo & student)
// postcondition: returns true if this student's credit hours ≥ 125
//                and GPA ≥ 2.0; otherwise, returns false
```

Part (c) begins on page 6.

GO ON TO THE NEXT PAGE

(c) Write function `FillSeniorList`, as started below. `FillSeniorList` determines which students in the array `roster` are seniors and copies those students' records to the array `seniors`. It should also set the value of parameter `numSeniors` to be the number of seniors in the array `seniors`.

In writing `FillSeniorList`, you may call function `IsSenior` specified in part (b). Assume that `IsSenior` works as specified, regardless of what you wrote in part (b).

Complete function `FillSeniorList` below. Assume that `FillSeniorList` is called only with parameters that satisfy its precondition.

```
void FillSeniorList(const apvector<StudentInfo> & roster,
                    int numStudents, apvector<StudentInfo> & seniors,
                    int & numSeniors)
// precondition: roster contains numStudents records,
//               0 < numStudents ≤ roster.length(),
//               and seniors is large enough to hold all of
//               the seniors' records
```

GO ON TO THE NEXT PAGE

ADDITIONAL WORKSPACE

GO ON TO THE NEXT PAGE