2. Periodically, a company processes the retirement of some of its employees. In this question, you will write functions to help the company determine whether an employee is eligible to retire and to process the retirement of all eligible employees.

The `Employee` class is declared as follows.

```
class Employee
{
  public:
    int Age() const;
    // returns the age (in years) of this employee

    int YearsOnJob() const;
    // returns the number of years this employee has worked

    double Salary() const;
    // returns the salary of this employee in dollars

    int ID() const;
    // returns unique employee ID number

  // ... constructors, other member functions and data not shown
};
```

The `Company` class is declared as follows.

```
class Company
{
  public:
    void ProcessRetirements();
    // postcondition: all retirement-eligible employees have been
    //                removed from empList; empList has been resized
    //                to reflect retirements;
    //                empList remains sorted by employee ID;
    //                salaryBudget has been updated to reflect remaining
    //                employees

    // ... constructor and other public methods not shown

  private:
    bool EmployeeIsEligible(const Employee & emp) const;
    // postcondition: returns true if emp is eligible to retire;
    //                otherwise, returns false

    apvector<Employee> empList;
    // empList.length() is the number of employees in this company

    int retireAge;           // minimum age to retire
    int retireYears;         // minimum years on job to retire
    double retireSalary;     // minimum salary to retire

    double salaryBudget;
    // total salary of all employees

};
```

GO ON TO THE NEXT PAGE.

The data member `empList` is sorted in ascending order by employee ID. The total of all salaries is maintained in the data member `salaryBudget`.

(a) An employee is eligible for retirement if (s)he meets at least two of the following requirements:

1. The employee is at least `retireAge` years old.
2. The employee has worked for at least `retireYears`.
3. The employee's salary is at least `retireSalary`.

Write the `Company` member function `EmployeeIsEligible`, which is described as follows. `EmployeeIsEligible` returns a Boolean value that indicates whether `Employee emp` is eligible for retirement, using the rules described above.

Complete function `EmployeeIsEligible` below.

```
bool Company::EmployeeIsEligible(const Employee & emp) const
// postcondition: returns true if emp is eligible to retire;
//                otherwise, returns false
```

(b) Write the `Company` member function `ProcessRetirements`, which is described as follows. `ProcessRetirements` removes all retirement-eligible employees from the `empList` array, resizes (shrinks) `empList` as appropriate (maintaining its order by employee ID), and decreases `salaryBudget` to reflect the salary of the remaining employees.

In writing `ProcessRetirements`, you may call `EmployeeIsEligible`, specified in part (a). Assume that `EmployeeIsEligible` works as specified, regardless of what you wrote in part (a).

Complete function `ProcessRetirements` below.

```
void Company::ProcessRetirements()
// postcondition: all retirement-eligible employees have been
//                removed from empList; empList has been resized
//                to reflect retirements;
//                empList remains sorted by employee ID;
//                salaryBudget has been updated to reflect remaining
//                employees
```

GO ON TO THE NEXT PAGE.