The materials included in these files are intended for use by AP teachers for course and exam preparation; permission for any other use must be sought from the Advanced Placement Program®. Teachers may reproduce them, in whole or in part, in limited quantities for noncommercial, face-to-face teaching purposes. This permission does not apply to any third-party copyrights contained herein. This material may not be mass distributed, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here.
(a) An employee is eligible for retirement if (s)he meets at least two of the following requirements:

1. The employee is at least \( \text{retireAge} \) years old.
2. The employee has worked for at least \( \text{retireYears} \).
3. The employee's salary is at least \( \text{retireSalary} \).

Write the \texttt{Company::EmployeeIsEligible} function, which is described as follows. \texttt{EmployeeIsEligible} returns a Boolean value that indicates whether \texttt{Employee emp} is eligible for retirement, using the rules described above.

Complete function \texttt{EmployeeIsEligible} below.

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

int x = 0;
if (emp.Age() >= retireAge)
  x++;
if (emp.YearsOnJob() >= retireYears)
  x++;
if (emp.Salary() >= retireSalary)
  x++;
return (x >= 2);
```

Part (b) begins on page 10.
(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.

```c
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

   for (x = 0; x < empList.length(); x++)
   
   if (EmployeeIsEligible(empList[x]))
   
   for (y = x; y < empList.length()-1; y++)
   
   empList[y] = empList[y+1];

   empList.resize(empList.length()-1);
   x--; 

   salaryBudget = 0;
   for (x = 0; x < empList.length(); x++)
   
   salaryBudget += empList[x].salary();

   return;
}
```

-----

*GO ON TO THE NEXT PAGE.*
(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.

```cpp
bool Company::EmployeeIsEligible(const Employee & emp) const
// postcondition: returns true if emp is eligible to retire; otherwise, returns false
{
   if (emp.Age() >= retireAge)
      if (emp.YearsOnJob() >= retireYears || emp.Salary() >= retireSalary)
         return(true);
   else return(false);
   if (emp.Salary() >= retireSalary)
      if (emp.YearsOnJob() >= retireYears ||emp.Age() >= retireAge)
         return(true);
   else return(false);
   if (emp.YearsOnJob() >= retireYears)
      if (emp.Salary() >= retireSalary || emp.Age() >= retireAge)
         return(true);
   else return(false); 
```

Part (b) begins on page 10.

**GO ON TO THE NEXT PAGE.**
(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.

```cpp
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
{
  vector<Employee> Valid(j);
  int k = 0;
  for (k = 0; k < empList.length(); k++)
    if (EmployeeIsEligible(empList[k])
      Valid[k] = empList[k];
      k++;
    Valid.resize(j+1);
  else
    salaryBudget -= empList[k].salary;

  empList = Valid;
}
```
(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.

```cpp
bool Company::EmployeeIsEligible(const Employee & emp) const
    // postcondition: returns true if emp is eligible to retire; otherwise, returns false
    
    if (emp.Age() >= retireAge) &&
        (emp.YearsOnJob() >= retireYears) &&
        (emp.Salary() >= retireSalary)
    
    return true;
    
    else
    
    return false;
```

Part (b) begins on page 10.
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.

```cpp
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

    for(int x = 0; x < empList.length(); x++)
    {
        if (EmployeeIsEligible(empList[x]))
        {
            salaryBudget = salaryBudget - empList[x].Salary();
            for(int y = x; y < empList.length(); y++)
            {
                empList[y] = empList[y+1];
            }
            empList.resize(empList.length()-1);
        }
    }
}
```

-10-

GO ON TO THE NEXT PAGE.