Feel free to use and modify **main.cpp** to test your function in **p1.hpp**.

You only need to complete the function definition in **p1.hpp**.

Problem 2

Template:

[p2 template.zip (https://bruinlearn.ucla.edu/courses/134192/files/10347468?wrap=1)](https://bruinlearn.ucla.edu/courses/134192/files/10347468?wrap=1) (https://bruinlearn.ucla.edu/courses/134192/files/10347468/download?download_frd=1)

In this problem, you will be provided with a template file that contains **main.cpp**, **Employee.hpp**, **Employee.cpp**, **Company.hpp**, **Company.cpp**.

About Employee.hpp

Private member variables and public member function declarations are provided.

Your task: complete two constructors.

1. Complete the default constructor. Set the default values of **name_** to be "No name", **salary_** to be 0, **age_** to be 0.

2. Complete the second constructor. Given three parameters name, salary, and age, set the member variables accordingly.

About Employee.cpp

You will complete 7 function definitions.

1. **set_name**, **set_salary**, **set_age** are mutators that replace member variables with a given parameter.

2. **get_name**, **get_salary**, **get_age** are accessors and pure functions that return member variables.

3. **print** function is an accessor that prints out the description of the Employee object.

About Company.hpp

Private member variables and public member function declarations are provided.

Your task: complete two constructors.

1. Complete the default constructor. Set the default values of **name_** to be "No name" and **vec_employees_** to be an empty vector.

2. Complete the second constructor. Given a single parameter name, set **name_** to be name and **vec_employees_** to be an empty vector.

About Company.cpp

You will complete 6 function definitions.

1. **get_name**: an accessor and a pure function that returns name_.

2. **get_employees**: an accessor and a pure function that returns vec_employees_.

3. **add_employee**: given an employee object, add it to the vector vec_employees_.

4. **sort_by_salary**: sort the vector vec_employees based on employees' salaries. It should sort them in descending order. (The employee with the highest salary should come first). [you can use the sort algorithm that was covered in W8-Monday lecture].

5. **increase_salary**: given a parameter rate, increase all employees' salaries by the given rate. For example, if rate = 0.2, then you should increase their salary by 20 %.

6. **print**: an accessor that prints out the description of the company.

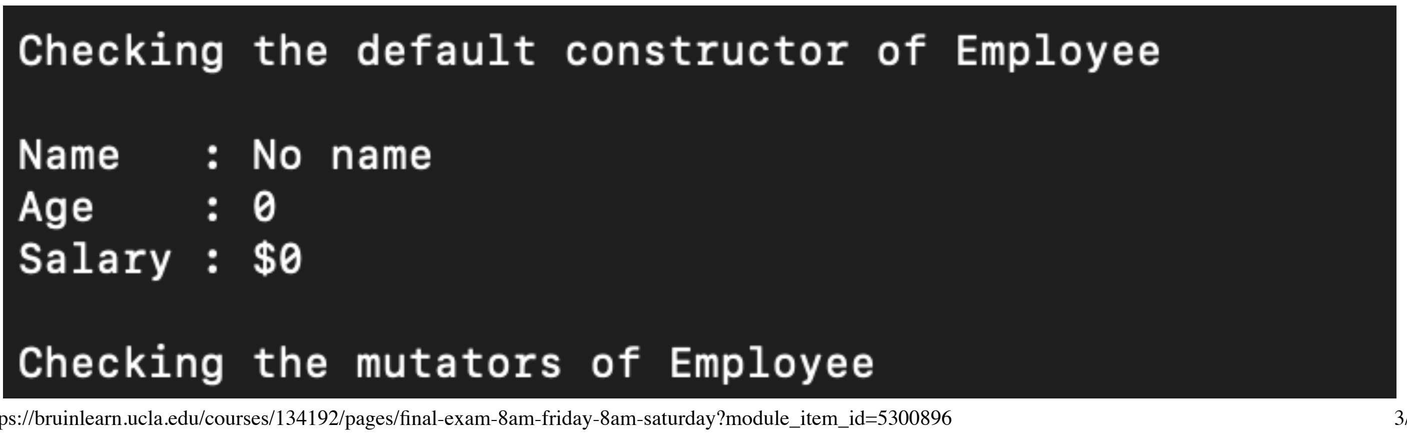
About main.cpp

main.cpp provides you with the test cases that we will use to test your codes.

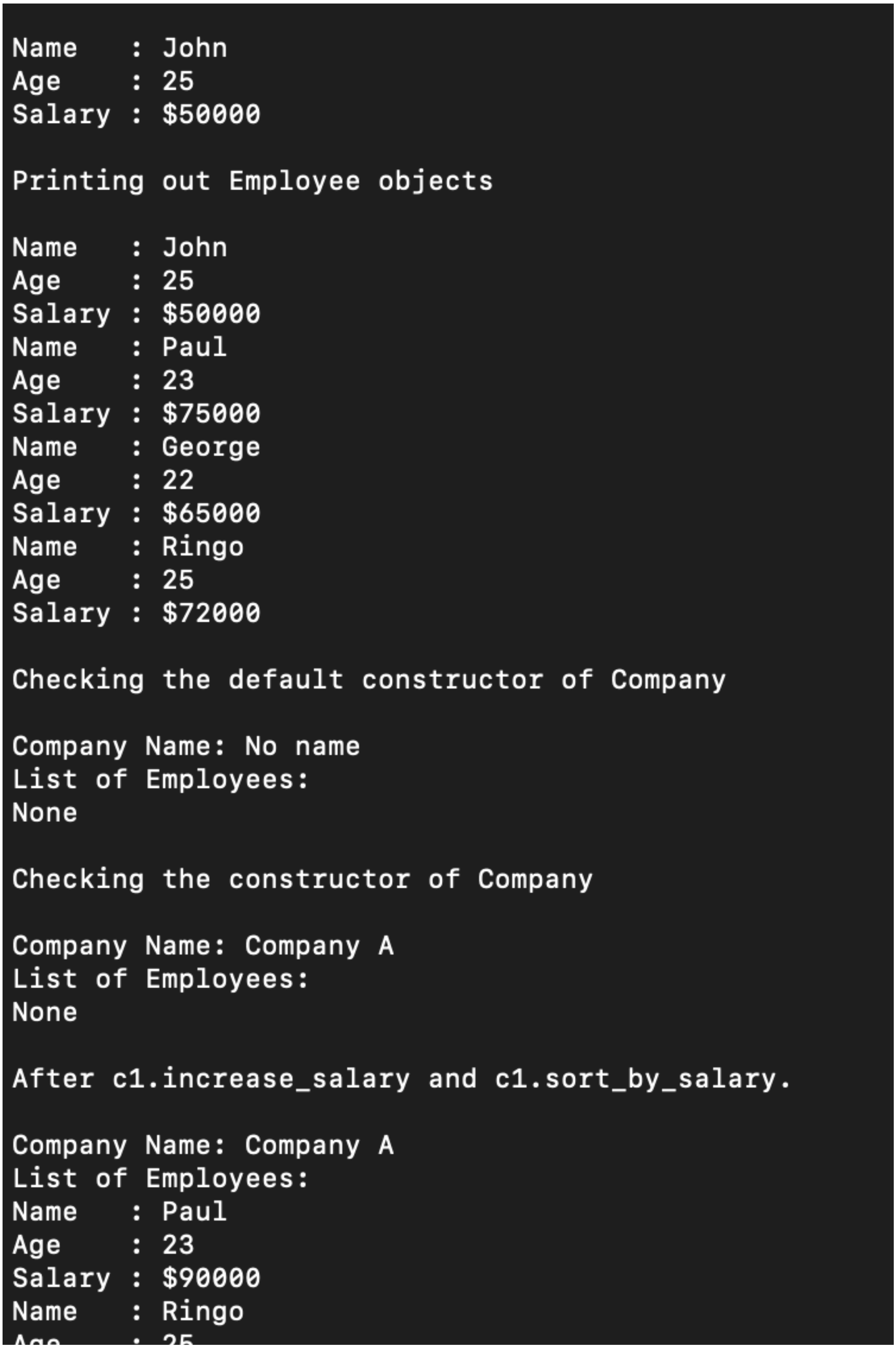
You do not need to do anything with the file.

Assuming that your codes work fine, your codes should successfully compile and run **main.cpp** file.

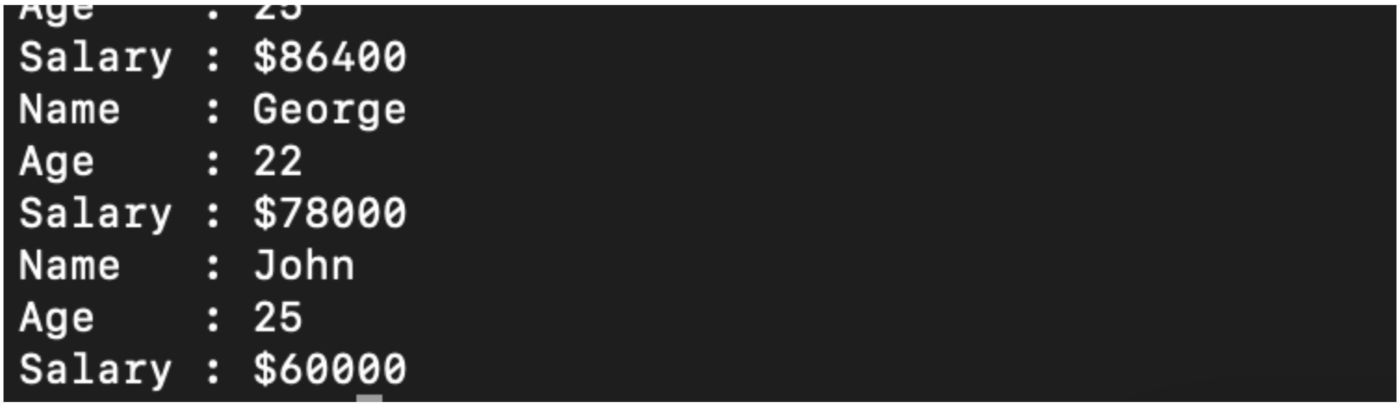
Here is the output that you should expect to see after you complete class definitions.



```
Checking the default constructor of Employee

Name   : No name
Age    : 0
Salary : $0

Checking the mutators of Employee
```



```
Name   : John
Age    : 25
Salary : $50000

Printing out Employee objects

Name   : John
Age    : 25
Salary : $50000
Name   : Paul
Age    : 23
Salary : $75000
Name   : George
Age    : 22
Salary : $65000
Name   : Ringo
Age    : 25
Salary : $72000

Checking the default constructor of Company

Company Name: No name
List of Employees:
None

Checking the constructor of Company

Company Name: Company A
List of Employees:
None

After c1.increase_salary and c1.sort_by_salary.

Company Name: Company A
List of Employees:
Name   : Paul
Age    : 23
Salary : $90000
Name   : Ringo
Age    : 25
```



```
Age    : 25
Salary : $86400
Name   : George
Age    : 22
Salary : $78000
Name   : John
Age    : 25
Salary : $60000
```