Lab Two

1. Document API for library

I have three classes in the library: Company, Employee and Task

1.1 Company

Attribute:

- int curEmployeeNum;
- int totalEmployeeNum;
- int taskId;
- vector<Employee> employees;
- vector<Task> tasks;

Method:

- void Company::Hire(int number)
- void Company::Fire(int id)
- void Company::createTask(int number)
- void Company::Report()

void Company::Hire(int number)

For this method, user will be asked to input an integer number. If the input number N is valid(an integer and greater than 0), N Employee objects will be added to employees vector, and method will output "N employees hired". Screenshot of the output of this method is shown below.

```
Choose operation: hire, fire, task, work, report, company, stop
hire
How many employees do you want to hire?
10
10 employees hired
```

void Company::Fire(int id)

For this method, user will be asked to input an integer number, If the input number N is valid(an integer and greater than 0), the employee whose id equals N will be removed out of employees vector, and method will output "Employee fired, id : N". If cannot find the employee whose id equals N, method will out put "Employee no found". Screenshot of the output of this method is shown below.

```
Choose operation: hire, fire, task, work, report, company, stop
fire
Which employee do you want to fire?
5
Employee fired, id : 5
```

void Company::createTask(int number)

For this method, user will be asked to input an integer number, If the input number N is valid(an integer and greater than 0), N Task objects will be added to tasks vector, and method will output "N tasks created". Screenshot of the output of this method is shown below.

```
Choose operation: hire, fire, task, work, report, company, stop
task
How many tasks do you want to create?
100
100 tasks created
```

void Company::Report()
There is not input required for this method. This method will output the total number of employees(including those fired), current employee number, number of remaining tasks and id of current employees. Screenshot of the output of this method is shown below.

```
Choose operation: hire, fire, task, work, report, company, stop
company
Total Employee Number: 10  Current Employee Number: 9  Remaining Tasks: 100
Current Employees: 1 2 3 4 6 7 8 9 10
```

1.2 Employee

Attribute:

- int id;
- int completedTaskNum;
- int workTime;
- vector<int> completedTask;

Method:

- void Employee::Report()

void Employee::Report()
There is no required input for this method. This method will output the work report of all current employees, including employee id, number of tasks this employee completed, total work time, and a list of id of completed tasks. Screenshot of the output of this method is shown below.

```
Choose operation: hire, fire, task, work, report, company, stop
report
Employee Id: 1 Number of Completed Tasks: 9 Work Time: 533
Tasks Completed: 100 85 81 77 66 55 43 27 6
Employee Id: 2 Number of Completed Tasks: 12 Work Time: 545
Tasks Completed: 99 87 84 83 78 71 49 36 31 9 5 2
Employee Id: 3 Number of Completed Tasks: 11 Work Time: 522
Tasks Completed: 98 88 86 67 48 35 34 21 19 10 8
Employee Id: 4 Number of Completed Tasks: 13 Work Time: 504
Tasks Completed: 97 90 70 62 47 45 33 29 26 18 16 12 1
Employee Id: 6 Number of Completed Tasks: 15 Work Time: 530
Tasks Completed: 96 80 69 60 59 53 51 50 42 30 23 20 17 11 4
Employee Id: 7 Number of Completed Tasks: 8 Work Time: 514
Tasks Completed: 95 82 64 58 41 32 25 15
Employee Id: 8 Number of Completed Tasks: 9 Work Time: 502
Tasks Completed: 94 76 74 72 56 40 37 22 7
Employee Id: 9 Number of Completed Tasks: 10 Work Time: 522
Tasks Completed: 93 91 75 61 57 52 44 38 28 13
Employee Id: 10 Number of Completed Tasks: 13 Work Time: 545
Tasks Completed: 92 89 79 73 68 65 63 54 46 39 24 14 3
```

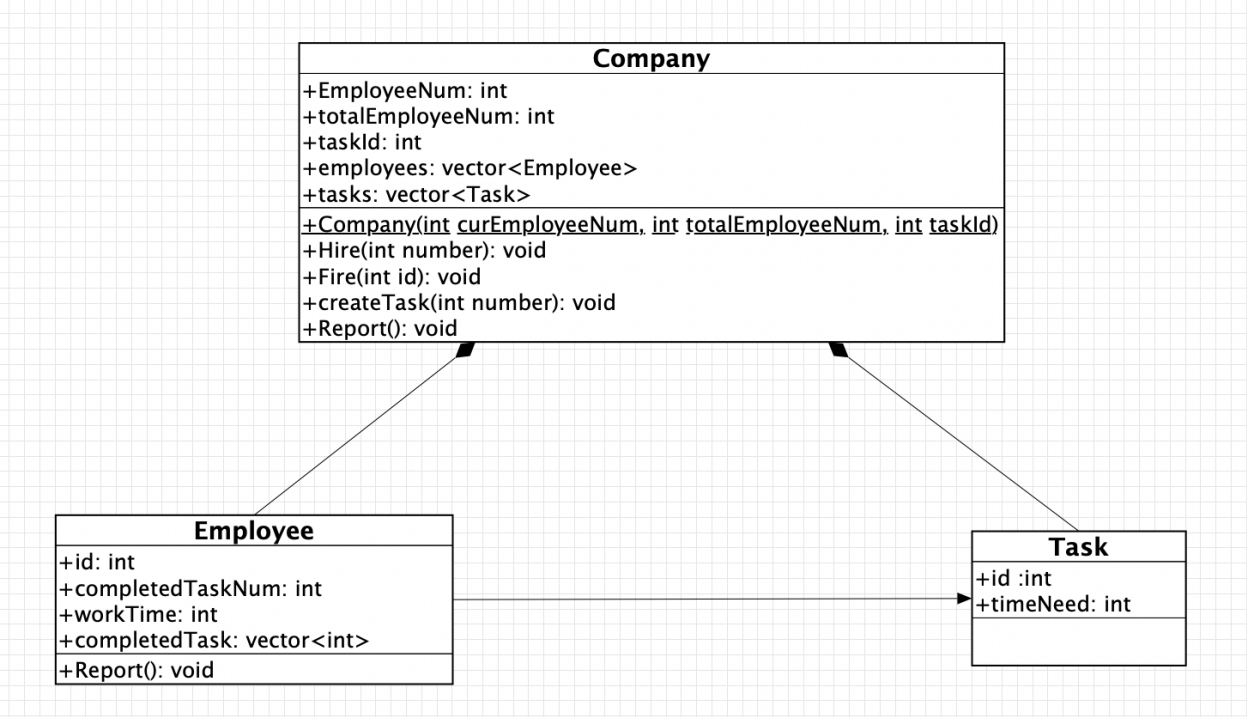
1.3 Task
Attribute:

- int id;
- int timeNeed;

1.4 Main
There is one function in main name work.

void work(Company *myCompany)
This function takes an pointer to a Company object and assign tasks to employees in the company. After assigning, the method will output "Current tasks completed". This method will look though all employees' total work time, and allocate the task to the one that have the shortest work time.

2. Design Concept
In this library, we have three classes: Company, Employee and Task. Singleton design pattern is used in this library. A company has multiple employees. Employee is a component of company. Therefore, relationship between class Company and class Employee is composition. A company has multiple tasks. Task is a component of company. Therefore, relationship between class Company and class Task in also composition. A task can only be assigned to one employee, and a employee can complete multiple tasks. Therefore, relationship between class Employee and class Task is association. Relationship between classes is shown in the UML below.



3. Compare Execution Time of the Library Code between Demo Application and Comparison Application

1. Paste 1000 execution runtime of demo application and comparison application to one excel file, and perform basic statistic measure on them.

Comparison Application	Demo Application
0.005877	0.066
0.005901	0.0619
0.005903	0.0513
0.005956	0.0503
0.006041	0.0524
0.005784	0.0502
0.005886	0.0515
0.005866	0.0696
0.005815	0.0624
0.005822	0.0603
0.005868	0.0522
0.005892	0.0499
0.005737	0.0494
0.005898	0.0484
0.00588	0.0523
0.005704	0.0521
0.005898	0.0502
0.005731	0.05
0.005994	0.049
0.005787	0.0488
0.00584	0.0501
0.005798	0.0493
0.00604	0.0488
0.005896	0.0641
0.006243	0.0638
0.005892	0.0529
0.005852	0.0494
0.005852	0.0531
0.005783	0.0479
0.005932	0.0495
0.005869	0.0482
0.005835	0.0494
0.006001	0.0494

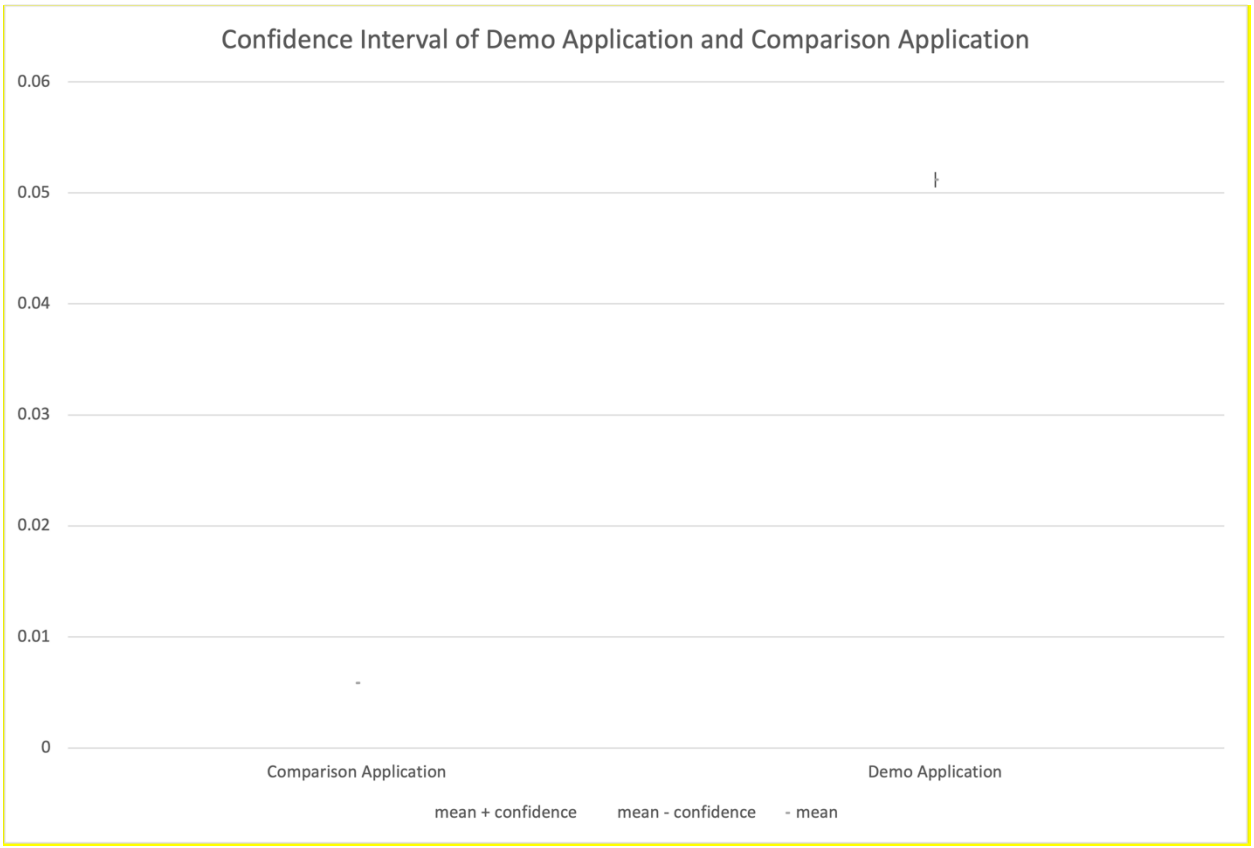
We can tell from the statistic results in the form below that execution time of comparison application is in the range of 0.005445 seconds to 0.006424 seconds, and that of demo application is in the range of 0.211 seconds to 0.0469 seconds. What is more, the average execution time of comparison application is 0.005830191 seconds, while the average execution time of demo application is 0.0511779 seconds, which is almost 10 times as that of native code base.

	Comparison Application	Demo Application
mean	0.005830191	0.0511779
max	0.006424	0.211
min	0.005445	0.0469

2. Use Excel to calculate 95% confidence intervals. We can see from the form below: 95% confidence interval of comparison application is 0.005822249 to 0.005838133, 95% confidence interval of demo application is 0.040087411 to 0.051865283.

	Comparison Application	Demo Application	WASM Improved
mean	0.005830191	0.0511779	0.0036056
max	0.006424	0.211	0.0149
min	0.005445	0.0469	0.0026
standard_dev	0.000128134	0.011090489	0.000881797
95% confidence	7.94165E-06	0.000687383	5.46533E-05
mean - confidence	0.005822249	0.050490517	0.003550947
mean + confidence	0.005838133	0.051865283	0.003660253

From the following graph we can tell that the lower bound of confidence interval of demo application is higher than the upper bound of confidence interval of comparison application, which means there is no overlap between confidence interval of demo application and comparison application. Therefore, we can conclude that these results are of statistically significant, performance of comparison application is much better than demo application.



4. Improve Performance of WASM Code

In native code base, I changed the remove the operator "+=" as shown in the screenshot below to improve the performance of wasm code.

```
// Orignal Code
// myCompany->employees[pos].workTime += task.timeNeed;


// Improve Performance
myCompany->employees[pos].workTime = myCompany->employees[pos].workTime + task.timeNeed;
```

Comparison Application	Demo Application	Improved
0.005877	0.066	0.0644
0.005901	0.0619	0.0648
0.005903	0.0513	0.0546
0.005956	0.0503	0.0554
0.006041	0.0524	0.0525
0.005784	0.0502	0.0485
0.005886	0.0515	0.0521
0.005866	0.0696	0.0544
0.005815	0.0624	0.0513
0.005822	0.0603	0.0723
0.005868	0.0522	0.0618
0.005892	0.0499	0.0494
0.005737	0.0494	0.0513
0.005898	0.0484	0.0534
0.00588	0.0523	0.059
0.005704	0.0521	0.0567
0.005898	0.0502	0.0575
0.005731	0.05	0.0506
0.005994	0.049	0.0483
0.005787	0.0488	0.0498
0.00584	0.0501	0.0486
0.005798	0.0493	0.0485
0.00604	0.0488	0.0483
0.005896	0.0641	0.0484
0.006243	0.0638	0.0508
0.005892	0.0529	0.0481
0.005852	0.0494	0.0487
0.005852	0.0531	0.0486
0.005783	0.0479	0.0491
0.005932	0.0495	0.0497

As the time comparison shown below, this change indeed improve the performance of wasm code. The mean of 1000 times running of after improvision is 0.01s faster than that before improvision.

	Comparison Application	Demo Application	Improved
mean	0.005830191	0.0511779	0.0507518
max	0.006424	0.211	0.1726
min	0.005445	0.0469	0.0475

2. Use Excel to calculate 95% confidence intervals. We can see from the form below: 95% confidence interval of comparison application is 0.005822249 to 0.005838133, 95% confidence interval of improved application is 0.05029541 to 0.05120819.

	Comparison Application	Demo Application	Improved
mean	0.005830191	0.0511779	0.0507518
max	0.006424	0.211	0.1726
min	0.005445	0.0469	0.0475
standard_dev	0.000128134	0.011090489	0.00736357
95% confidence	7.94165E-06	0.000687383	0.00045639
mean - confidence	0.005822249	0.050490517	0.05029541
mean + confidence	0.005838133	0.051865283	0.05120819

From the following graph we can tell that the lower bound of confidence interval of improved application is higher than the upper bound of confidence interval of comparison application, which means there is no overlap between confidence interval of improved application and comparison application. Therefore, we can conclude that these results are of statistically significant, performance of comparison application is much better than improved application.