OOP, Spring 2015

Final Exam

Exam rules:

1. **Exam time: 18:30 PM ~ 21:30 PM**
2. **Do not use any communication device such as mobile phone, iPad, laptop … etc.**
3. **Do not use any removable storage device, such as USB …**
4. **This exam has 3 questions. The scoring of each question is listed as follows: 30% for Question1, 40% for Question2, 50% for Question3.**
5. **Do not use any global variable in this exam.**
6. **You can append any member function in all classes.**
7. **Your code must be executed correctly without any warning message, compiling error, and runtime error.**
8. **Upload all submissions to E3.**

Problem 1

Description

There is a text file including employee’s name and full-time or part-time in the first line. And the next four lines are the hour employee worked in one month. You need to calculate employee’s salary in that month. For full-time salary, employee should work 8 hours a day or salary would discount 20% in one day. If working hour over 8 hours a day then employee would receive NT$200 bonus in one day. On weekend(Sat. and Sun.) working hours should not exceed 4 hours a day, and it would get NT$300/hour in one day. For part-time salary, it depends on the working hours. But if employee absence from duty, it would deduct NT$500 in one day.

Requirement

1. **File I/O: Read the text file “workinghours.txt”** and output your result to the file named “salary.txt”.
2. Base class named Employee, derived class named Part-time and Full-time. Employee’s name, working hours and salary only in the Employee class.
3. Use **polymorphism** and **virtual function** on function **pay** and **show**. Function pay to calculate salary, function show to output the result.

Sample I/O

Sample input(Read from “workinghours.txt”)

```
Bob part-time
Mon.  Tue.  Wed.  Thu.  Fri.  Sat.  Sun.
 4     3     4     0     8     5     7
 6     2     5     2     0     0     0
 4     4     2     2     4     4     4
 0     0     0     4     4     6     7
 4     5

Josh full-time
Mon.  Tue.  Wed.  Thu.  Fri.  Sat.  Sun.
 4     3     4     0     8     5     7
 6     2     5     2     10    0     0
 4     4     2     2     4     4     4
```

```
0    0    0    4    4    6    7
4    5
```

Sample output(Write to “salary.txt”)

```
Bob part-time
Working hour:100 hours
Absence:7 days
Hourly wage:120
Salary: NT$8500

Josh full-time
Working hour:110 hours
Absence:7 days
Overtime hours:24 hours
Daily wage:2000
Salary: NT$37000
```

Problem 2

Description

In this problem, you have to use function template to implement three functions, those are “Add”, “Sub”, and “Mul”. Those functions have two parameters and return the result of adding, subtracting, and multiplying respectively.

Requirement

1. Write the class named Vector. (You just have to handle the condition of 2-dimentional.)
2. Write the class named Matrix. (You just have to handle the condition of square matrix.)
3. Use function template to implement the functions mentioned above, which can handle the type of parameters are both integers, or doubles, or Matrixes, or Vectors at the same time. (Hint: You can use the operator overloading to solve this problem.)
4. The multiplication of two Vectors means the Vector1 dot Vector2.

Input

1. The first line indicates the values int_num1 and int_num2.
2. The second line indicates the values of double_num1 and double_num2.
3. The third and forth lines indicate the content of Vector1 and Vector2 respectively.
4. The fifth line indicates the size of Matrix1 and Matrix2.
5. The following two parts indicate the content of Matrix1 and Matrix2.

Output

(1) The result of the Add of two integer numbers.
(2) The result of the Sub of two integer numbers.
(3) The result of the Mul of two integer numbers.
(4) The result of the Add of two double numbers.
(5) The result of the Sub of two double numbers.

(6) The result of the Mul of two double numbers.

(7) The result of the Add of two Vectors.

(8) The result of the Sub of two Vectors.

(9) The result of the Mul of two Vectors.

(10) The result of the Add of two Matrixes.

(11) The result of the Sub of two Matrixes.

(12) The result of the Mul of two Matrixes.

Sample Input	Sample Output
//Input int numbers	Integer part:
100 200	300
//Input double numbers	-100
24.81 6.23	20000
//Input content of vectors	Double part:
333 444	31.04
111 222	18.58
//Input size of matrixes	154.5663
3	Vector part:
//Input content of matrixes	444 666
1 1 1	222 222
1 1 1	135531
1 1 1	Matrix part:
	2 2 2
1 1 1	2 2 2
1 1 1	2 2 2
1 1 1	
	0 0 0
	0 0 0
	0 0 0
	3 3 3
	3 3 3
	3 3 3

Problem 3

Template

Description:

You need to implement Vector which size is changeable.

Requirement:

- Forbidden include <vector>.
- At fisrt,the vector has no space for data. fixed large enough space for data in the sampling test is forbidden.
- Everytime you store new data you need to check whether the vector is full .If so, double its space.
- When you pop last data in your-defined vector , you need to check whether its data size is less than half of container size. If so, decrease half of space.
- Your-defined vector should support many data type.
- Sort the sampling test.

- Please read data in the file(input.txt).
- Please output a file([Student_ID]_output.txt).

Input.txt
9 7 3 5 19 23 -5 -7 -8 10 11 0 6 17 end
-2.5 5.3 3.6 7.0 -11.2 end
00 doraemon 01 snoopy 03 cat 07 dog end

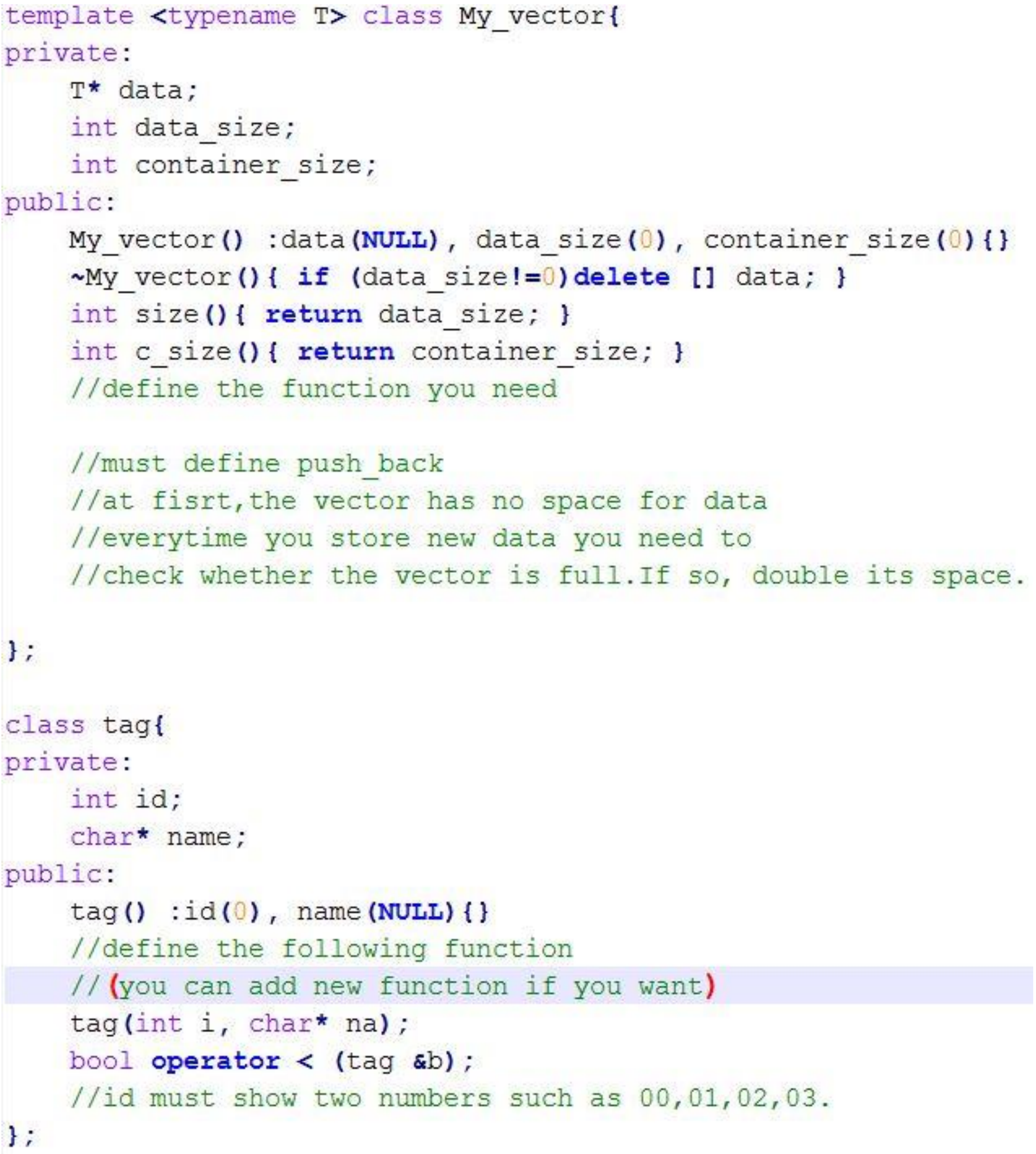
Don’t push “end” in your-defined vector.

In the last sample test : number + string treat as a data.

[Student_ID]_output.txt	comment
-8 -7 -5 0 3 5 6 7 9 10 11 17 19 23 (14/16)	//sort
after pop 7 times: -8 -7 -5 0 3 5 6 (7/8)	// pop
-11.2 -2.5 3.6 5.3 7 (5/8)	//sort
after pop 2 times:-11.2 -2.5 3.6 (3/4)	//pop
03 cat 07 dog 00 doraemon 01 snoopy (4/4)	// sort by name
after pop 2 times: 03 cat 07 dog (2/4)	//pop

The last (?/?) for each line means (data size/ container size).

Appendix A



```
template <typename T> class My_vector{
private:
    T* data;
    int data_size;
    int container_size;
public:
    My_vector() :data(NULL), data_size(0), container_size(0){}
    ~My_vector(){ if (data_size!=0)delete [] data; }
    int size(){ return data_size; }
    int c_size(){ return container_size; }
    //define the function you need

    //must define push_back
    //at fisrt,the vector has no space for data
    //everytime you store new data you need to
    //check whether the vector is full.If so, double its space.

};

class tag{
private:
    int id;
    char* name;
public:
    tag() :id(0), name(NULL){}
    //define the following function
    //(you can add new function if you want)
    tag(int i, char* na);
    bool operator < (tag &b);
    //id must show two numbers such as 00,01,02,03.
};
```