COMP2123B Programming Technologies and Tools
Quiz

- **Submit your work to Moodle VPL**, for each question, we will manually check your program if the VPL doesn't work for your program.
- **If the server is too slow, you may write your answer to the given blank paper.** We will grade it manually and focus on logic not exact syntax. If you also submit through VPL, **we will use the max score.**
- Please download **`Quiz.tar`** and extract the files.
- Please put your student ID card in front of the computer.

Question 1 [40%]. Find the most popular book borrowed.
The HKU Library service has installed smart card scanners in the book borrowing machines. All scanners are connected to a central backend system that stores the access records of students. The system generates a log file every month, each line in the log file represents a transaction made through the book borrowing machines and has the following format:

```
[Timestamp],[StudentID],[Book name],[Borrow/Return]
```

- `Timestamp` has the format `YYYY-MM-DD_HH:MM:SS.Millisecond`.
- `StudentID` is an 6-digit value identifying a student.
- `Book name` is the name of a book; it may consist of spaces.
- `Borrow/Return` represents whether the student is borrowing or returning a book.
- Comma "`,`" is used as field separator in the log file.
- The log files of the HKU Library have "`Library_`" as prefix in their filenames, followed by the date in `YYYY-MM`, followed by "`.log`".
- You can assume that the transactions are sorted in ascending order of the Timestamps.
- Note that your script will be graded with additional test cases.

In directory `Q1`, there are files with "`.log`" as suffix and a shell script `stat.sh`. Your task is to update the shell script `stat.sh`, so that if we call the script and pass the timestamp prefix *`t`* as **the first input argument**:

```
./stat.sh t
```

- The script considers only the access logs with ***`t`* as the prefix of timestamp, and those that are "Borrow" entries**. **Hints:** match pattern from the start of the string (but not the end).
- **[40%]** The script displays the name of books borrowed by **the most number of distinct students**, and the corresponding distinct students count. Each line of the output should be in the following format.
  ```
  [number of distinct students] [book name]
  ```
  Note that spaces before/after/between the two values will be ignored when grading.
- **[40%]** If there are multiple books borrowed by the most number of distinct students, output in descending lexicographical order of Book name.
- **[20%]** Output "`No records found`" if there are no book to output.

Sample input and output

Suppose there are two log files `Library_2019-01.log`, and `Library_2019-02.log`

`Library_2019-01.log`:

```
2019-01-14_02:38:40.109530,78368999,Dubliners,Borrow
2019-01-15_01:05:12.820566,23632160,The Iliad,Borrow
2019-01-17_02:34:00.035900,89147871,Frankenstein,Borrow
2019-01-17_23:20:29.469121,21051061,Peter Pan,Borrow
2019-01-20_02:45:03.991528,34317899,The Iliad,Borrow
2019-01-21_16:46:28.260751,78368999,Dubliners,Return
2019-01-22_07:10:01.339826,56894553,A Modest Proposal,Borrow
2019-01-23_10:03:38.228600,34317899,Ulysses,Return
2019-01-23_20:42:56.713945,34317899,A Tale of Two Cities,Return
2019-01-30_07:30:38.846204,21051061,Peter Pan,Return
```

`Library_2019-02.log`:

```
2019-02-07_03:11:02.354281,23632160,The Iliad,Return
2019-02-09_05:39:41.749631,56894553,Metamorphosis,Return
2019-02-09_16:30:55.700732,34317899,The Iliad,Return
2019-02-14_11:46:01.725389,89147871,Frankenstein,Return
2019-02-14_16:38:29.048608,23632160,The Iliad,Borrow
2019-02-16_15:44:59.720611,56894553,Frankenstein,Borrow
2019-02-21_00:13:14.521378,56894553,Peter Pan,Borrow
2019-02-23_04:39:55.936762,23632160,The Iliad,Return
2019-02-27_17:50:17.690604,56894553,Frankenstein,Return
2019-02-28_13:01:45.329141,45968903,Pride and Prejudice,Borrow
```

Some sample execution of the script and the corresponding output is showed below. input is highlighted.

```
$ ./stat.sh 2019-01-14_02:38:40.109530
1 Dubliners
$ ./stat.sh 2019-01-1
1 The Iliad
1 Peter Pan
1 Frankenstein
1 Dubliners
$ ./stat.sh 2019-01
2 The Iliad
$ ./stat.sh 2019-0
2 The Iliad
2 Peter Pan
2 Frankenstein
$ ./stat.sh 2018
No records found
```

Question 2 [40%]. Quaternion

A **quaternion** is a representation commonly used in 3D computer graphics calculation. It can be represented using the form:

$$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

where ***a***, ***b***, ***c*** and ***d*** are real numbers; **`i`**, **`j`** and **`k`** represent three different dimensions in the quaternion number system.

Implement a class `Quaternion` that represents a quaternion. You have complete freedom on the design and implementation of the class, except that your class needs to satisfy the following requirements.

- All quaternion to be stored can be represented using 4 values, `a`, `b`, `c`, and `d`.
- You need to provide a header file `Quaternion.h` and an implementation file `Quaternion.cpp`.

A template for `Quaternion.h` and `Quaternion.cpp` is provide for you in directory `Q2`. Sample client programs are also provided for your testing, please note that your work will be graded with additional client programs.

Please implement the following operations.

- **[5%] Constructors.**
  Implement the following two constructors:
  i. `Quaternion()`:
     A default **constructor** which initializes the value of an object to `0+0i+0j+0k`.
  ii. `Quaternion(double a, double b, double c, double d)`:
     A constructor which initializes the value of an object to `a+bi+cj+dk`.

  Sample Client 1

```
int main(){
    Quaternion q1;
    Quaternion q2(1,2,3,4);
    return 0;
}
```

  Output: *nil*

  (Two quaternion objects representing the values $0 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$ and $1 + 2\mathbf{i} + 3\mathbf{j} + 4\mathbf{k}$ should be created without error messages, there will be no output for this test case)

- **[35%]** The class `Quaternion` can be printed directly using the **insertion** `operator(<<)`.
  - When an object is printed, it is printed in the form of `a+bi+cj+dk` .
  - There should be no space before and after "+".
  - The values of `a`, `b`, `c` and `d` must be printed with 1 decimal place. E.g., the value $1.26 + 2\mathbf{i} - 0.6\mathbf{j} + 1.33\mathbf{k}$ should be printed as `1.3+2.0i-0.6j+1.3k`.
    Hint: To print double type variable, say, *`myFloat`*, in 1 decimal place, you may include the library iomanip by "`#include <iomanip>`" then print it with: `cout << fixed << setprecision(1) <<` *`myFloat`* `<<...`

Sample Client 2

```
int main(){
    Quaternion q1;
    cout << q1;
   return 0;
}
```

Output: `0.0+0.0i+0.0j+0.0k`

Sample Client 3

```
int main(){
    Quaternion q1(1,2,3,4);
    cout << q1;
    return 0;
}
```

Output: `1.0+2.0i+3.0j+4.0k`

Sample Client 4

```
int main(){
    Quaternion q1(-1,-2,-3,-4);
    cout << q1;
    return 0;
}
```

Output: `-1.0-2.0i-3.0j-4.0k`

Sample Client 5

```
int main(){
    Quaternion q1(1.26,2,-0.6,1.33);
    cout << q1;
    return 0;
}
```

Output: `1.3+2.0i-0.6j+1.3k`

Sample Client 6

```
int main(){
    Quaternion q1(1,2,3,4);
    cout << q1 << q1;
    return 0;
}
```

Output: `1.0+2.0i+3.0j+4.0k1.0+2.0i+3.0j+4.0k`

- **[30%]** The class `Quaternion` should support the `+` operator and return a `Quaternion` object with value equals to the result of such operation. The operators are applied as if the quaternion is a polynomial with variables i, j, and k.

$$(a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}) + (a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) \\ = (a_1 + a_2) + (b_1 + b_2)\mathbf{i} + (c_1 + c_2)\mathbf{j} + (d_1 + d_2)\mathbf{k}$$

Sample Client 7

```
int main(){
    Quaternion q1(1, 2, 3, 4);
    cout << q1 + q1;
    return 0;
}
```

Output: `2.0+4.0i+6.0j+8.0k`

Sample Client 8

```
int main(){
    Quaternion q1(1, 2, 3, 4);
    Quaternion q2(-1, 2, -3, 4);
    cout << q1 + q2;
    return 0;
}
```

Output: `0.0+4.0i+0.0j+8.0k`

Sample Client 9

```
int main(){
    Quaternion q1(1, 2, 3, 4);
    cout << q1 + q1 + q1;
    return 0;
}
```

Output: `3.0+6.0i+9.0j+12.0k`

- **[30%]** The class `Quaternion` should support the post-increment operator `++`.

Sample Client 10

```
int main(){
    Quaternion q1(1, 2, 3, 4);
    cout << q1++ << q1;
    return 0;
}
```

Output: `1.0+2.0i+3.0j+4.0k2.0+3.0i+4.0j+5.0k`

Question 3 [20%]. Makefile

Many programs need to manipulate fractions like ½, ¾, etc. In directory `Q3`, you can find an implementation of a class `Fraction` in files `Fraction.h` and `Fraction.cpp`, as well as a client program `sum.cpp`. The `Fraction` class provides the following features.

- It stores a fraction by two integers `numerator` and `denominator`, which are both private members.
- It contains two constructors `Fraction()` and `Fraction(int num, int den)`.
- It supports the addition operator (i.e., +).
- It contains a function `print()` which returns a string corresponding to the value.

Your task is to create a `Makefile` with the following target. You may include other targets if needed.

- **`sum`**:
  - "`make sum`" should build the executable **`sum`**.
  - It should recompile the minimum number of files needed. For example, if none of the source files is modified, it will not recompile anything.
- **`backup`**:
  - "`make backup`" should create a directory `bak/` and copy `sum.cpp`, `Fraction.h`, and `Fraction.cpp` to the `bak/` directory. You should ensure that `bak/` will not contain any other files or directories.
  - "`make backup`" should work correctly even if a file with name `backup` exist.

- END OF QUIZ PAPER -