**CS F213 Object Oriented Programming**

**Minor Assignment 3**

**General instructions for all questions:**

* ***Your output should not have any leading or trailing whitespace(s) or newline character(s).***
* **More offline test cases will be used for the final grading.**
* **On the VPL interface on Moodle:**
  + **Go to the question**
  + **You can either go to the Submission tab or the Edit tab. The Submission tab lets you upload a program that you have written offline. The Edit tab lets you use the in-built IDE to write your program**
  + **In both cases, you can click the run / evaluate button to run your program and test it against the online test cases**
  + **The interface will tell you how many test cases have passed**
  + **The main method should be in a public class. The name of the public class should be the same as the Java filename**
  + **The number of submissions will not be an evaluation criteria**
  + **The execution time will be an evaluation criteria**
* **The anti-plagiarism instructions remain the same for the course.**

**Question 1**

* Create a Generic Java class called Sorter which can take an array of Integers, Doubles or Strings in the constructor(s).
* Write Lambda Expressions (block or single line) to define sorting for the array defined above:
  + Sorting order is: Dictionary order for Strings, and ascending order for Integers or Doubles.
  + You may use any sorting algorithm(s). Execution time will be an evaluation criteria. You are not allowed to use any sorting functions from pre-existing libraries.
* Pass the Lambda Expression as an argument to a Sorter.sort() method to execute the sorting. There should be only one sort() method in the Sorter class.
* The Comparator interface is not allowed. String comparison methods in the String class are not allowed. Method and Constructor references are not allowed.

**Input:**

A filename will be passed as a commandline argument to the main method. The file will have the type of data mentioned in the first line. In the second line the file will contain a space separated list of literals of the specified data type (all Strings will be one-word long)

*Example 1:*

Integer

3 90 8 26 17

*Example 2:*

String

BITS Hyderabad pilani dubai goa mumbai OOP

**Output:**

A space separated list of the sorted literals based on the sorting criteria for the data type. The output should not contain any leading or trailing whitespaces.

*Example 1:*

3 8 17 26 90

*Example 2:*

BITS dubai goa Hyderabad mumbai OOP pilani

Minimum number of items in the list: 1

Maximum number of items in the list: 1000000

**Question 2**

Use Java to solve the following. You may use String libraries to parse the input file:

* Create a **record** called Student with the following fields: ID (int), Name (String), Programme (Enum), CGPA (double)
* Create an enumeration called Programme with the following constants: CSIS, EEE, Mechanical, Chemical, Civil, Maths, Biology, Physics, Chemistry, Pharmacy, EcoFin, HSS. This enumeration will be used in the Student Record (above) for the Programme field
* Maintain an array of Student records in ascending order of ID
* Create a static class StudentRecordOps with methods to support the following commands:
  + insert <id> <name> <programme> <cgpa>
    - inserts a new record in the array in the correct position of ascending order of ID
    - Duplicate IDs are not allowed and must be handled using a custom exception
    - On successful insertion, the programme prints ‘Insert OK id: <id>’(without quotes) on the console (see example below).
    - Method signature: StudentRecordOps.insert(Student[] s\_arr, Student new\_student);
  + delete <id>
    - Deletes the record with the specified ID and defragments the array (shifts all the successive records to fill the space of the deleted array)
    - Deletion of non-existent IDs must be handled using a custom exception
    - On successful deletion, the programme prints ‘Delete OK id: <id>’(without quotes) on the console (see example below).
    - Method signature: StudentRecordOps.delete(Student[] s\_arr, int id);
  + display <id>
    - Displays the record with the given id in a space separated format
    - Display of non-existent IDs must be handled using a custom exception (see example below)
    - Method signature: StudentRecordOps.display(Student[] s\_arr, int id);
    - The CGPA should be displayed upto two decimal places using the String.format( "%.2f", dub ) method.
  + stats
    - Displays the total number of records and the average CGPA (upto 2 decimal places) of all the records, in that instant (average is 0.00 when there are no records). The two values are displayed according to the format in the example below
    - Method signature: StudentRecordOps.stats(Student[] s\_arr);
    - The average CGPA should be displayed upto two decimal places using the String.format( "%.2f", dub ) method.
  + save <filename>
    - Serialises the array of records and saves it to a file identified by the given filename (overwrite if file exists)
    - After a successful save, the program displays on the console ‘Save OK filename: <filename>’ without the quotes
    - Method signature: StudentRecordOps.save(Student[] s\_arr, String filename);
  + dump
    - Dump the contents of the entire array, one record per line
    - Method signature: StudentRecordOps.dump(Student[] s\_arr);
    - The dump() method should use the display method
    - The CGPA should be displayed upto two decimal places using the String.format( "%.2f", dub ) method.

**Input:**

A filename is passed on the command line. The file contains one command (from among the commands listed above) per line. Maximum number of commands in a file: 1000.

*Example:*

insert 1 Sunil CSIS 8.4

insert 3 Alex EEE 9.0

insert 2 Megha Mechanical 7.0

dump

delete 3

delete 3

insert 1 Faizal Physics 8.9

stats

display 1

**Output:**

The results of each command, one per line, without any trailing or leading whitespaces.

*Example:*

Insert OK id: 1

Insert OK id: 3

Insert OK id: 2

1 Sunil CSIS 8.40

2 Megha Mechanical 7.00

3 Alex EEE 9.00

Delete OK id: 3

Exception: Non-existent ID 3

Exception: Duplicate ID 1

#records: 2; avg CGPA: 7.70

1 Sunil CSIS 8.40